**FINAL PROJECT – Analisis Data Statistik A**

# EVALUASI, PERBANDINGAN, DAN PERAMALAN MODEL ARIMA DAN LSTM: STUDI KASUS KONSUMSI MINYAK BRASIL BERBASIS DATA NON-LINEAR


**Disusun Oleh:**

Joycelin Gracelda Resi Gaya 5003221121

Anisa Ardiani Putri 5003221171

**Dosen Pengampu:**

Dr. Irhamah, S.Si., M.Si.

NIP 19780406 200112 2 002

**Program Studi Statistika**

**Fakultas Sains dan Analitika Data**

**Institut Teknologi Sepuluh Nopember Surabaya**


**2025**

**FINAL PROJECT – Analisis Data Statistik A**

# EVALUASI, PERBANDINGAN, DAN PERAMALAN MODEL ARIMA DAN LSTM: STUDI KASUS KONSUMSI MINYAK BRASIL BERBASIS DATA NON-LINEAR


**Disusun Oleh :**

Joycelin Gracelda Resi Gaya 5003221121

Anisa Ardiani Putri 5003221171

**Dosen Pengampu:**

Dr. Irhamah, S.Si., M.Si.

NIP 19780406 200112 2 002

**Program Studi Statistika**

**Fakultas Sains dan Analitika Data**

**Institut Teknologi Sepuluh Nopember Surabaya**


**2025**

# ABSTRAK

Minyak bumi masih menjadi sumber energi utama di Brasil dan memegang peranan penting dalam mendukung pembangunan ekonomi negara. Prediksi produksi energi minyak yang akurat sangat diperlukan untuk perencanaan energi nasional yang efektif. Penelitian ini membandingkan kinerja model ARIMA (Autoregressive Integrated Moving Average) dan LSTM (Long Short-Term Memory) dalam meramalkan produksi energi minyak di Brasil yang diukur dalam satuan GWh. Data harian produksi minyak dianalisis menggunakan pendekatan time series, dimulai dari pengecekan kelengkapan data, transformasi menjadi data stasioner melalui differencing, hingga pemodelan dan evaluasi performa prediksi. Hasil penelitian menunjukkan bahwa model LSTM mampu menangkap pola non-linier lebih baik dibandingkan ARIMA, namun ARIMA tetap efektif untuk pola linier pada data. Evaluasi menggunakan metrik RMSE, MAE, dan MAPE menunjukkan keunggulan relatif masing-masing model. Temuan ini diharapkan dapat menjadi referensi bagi pengambil kebijakan dan praktisi energi dalam memilih metode peramalan yang tepat untuk data energi minyak di masa depan.

**Kata Kunci : ARIMA, LSTM, Time series, Energi Minyak, Prediksi, Brasil, Differencing, Stasionerita**

# DAFTAR ISI

**ABSTRAK**.....i
**BAB I PENDAHULUAN**.....5
1.1 Latar Belakang.....5
1.2 Rumusan Masalah.....5
1.3 Tujuan.....6
1.4 Manfaat.....6
**BAB 2 TINJAUAN PUSTAKA**.....7
2.1 Teori Tine Series.....7
2.2 Model ARIMA (Autoregressive Integrated Moving Average).....7
2.3 Pengujian Non Linearitas.....12
2.4 Deep Learning.....12
2.5 Jaringan Syaraf Tiruan/Neural Network (NN).....13
2.6 Model LSTM (Long Short Term Memory).....13
2.7 Evaluasi Model Time Series.....15
2.8 Oil Production Brasil.....16
2.9 Google Colab.....16
2.10 R-Studio.....17
**BAB 3 METODOLOGI PENELITIAN**.....19
3.1 Sumber Data.....19
3.2 Variabel Penelitian.....19
3.3 Tahapan Analisis.....19
3.4 Diagram Alir.....20
**BAB 4 HASIL DAN PEMBAHASAN**.....23
4.1 Karakteristik Data GWh Minyak di Brasil.....23
4.2 Uji Non Linearitas Data.....24
4.3 Hasil Pre-Processing ARIMA.....25
4.4 Hasil dan Analisis Model ARIMA.....26
4.5 Hasil dan Analisis Model LSTM.....28
4.6 Perbandingan Kinerja Model.....30
**BAB 5 KESIMPULAN DAN SARAN**.....32
5.1 Kesimpulan.....32
5.2 Saran.....32
**DAFTAR PUSTAKA**.....34
**LAMPIRAN**.....35

# DAFTAR GAMBAR

**Gambar 2.1** Penjelasan Tingkatan Deep Learning.....10
**Gambar 2.2** Arsitektur LSTM.....13
**Gambar 2.3** Forget Gates LSTM.....14
**Gambar 2.4** Input Gate dan tanh Layer pada LSTM.....14
**Gambar 2.5** Pembaharuan Cell pada LSTM.....15
**Gambar 2.6** Output Gate pada LSTM.....15
**Gambar 3.1** Diagram Alir ARIMA.....21
**Gambar 3.2** Diagram Alir LSTM.....22
**Gambar 4.1** Plot Data Time Series.....23
**Gambar 2. 1** Penjelasan Tingkatan Deep Learning.....12
**Gambar 2. 2** Arsitektur LSTM.....13
**Gambar 2. 3** Forget Gates LSTM.....14
**Gambar 2. 4** Input Gate dan tanh Layer pada LSTM.....15
**Gambar 2. 5** Pembaharuan Cell pada LSTM.....15
**Gambar 2. 6** Output Gate pada LSTM.....16
**Gambar 3. 1** Diagram Alir ARIMA.....21
**Gambar 3. 2** Diagram Alir LSTM.....22
**Gambar 4. 1** Plot Data Time Series.....23
**Gambar 4. 2** Gambar ACF data Oil Production Brazil.....25
**Gambar 4. 3** Gambar PACF data Oil Production Brazil.....25
**Gambar 4. 4** Gambar Perbandingan Forecast dan Data Test.....26
**Gambar 4. 5** Gambar Hasil Forecast 30 Hari Kedepan.....27
**Gambar 4. 6** Visualisasi Hasil.....30

## DAFTAR TABEL

**Tabel 2.1** Struktur Plot ACF dan PACF pada Model ARIMA.....12
**Tabel 3.1** Variabel Penelitian.....19
**Tabel 4.1** Hasil Uji Non Linearitas.....21
**Tabel 3. 1** Variabel Penelitian ..... 19
**Tabel 4. 1** Hasil Uji Non Linieritas (Teraesvirta Neural Network Test)…….24
**Tabel 4. 2** Hasil Uji Stationer ADF Data Asli ..... 24
**Tabel 4. 3** Hasil Uji Stationer Setelah Transformasi ..... 24
**Tabel 4. 4** Model ARIMA ..... 26
**Tabel 4. 5** Evaluasi Model ARIMA..... 26
**Tabel 4. 6** Kombinasi Parameter Terbaik dengan Tuning..... 28
**Tabel 4. 7** Hasil Pelatihan Model ..... 29
**Tabel 4. 8** Perbandingan Akurasi Model ARIMA dan LSTM ..... 31

# BAB 1
# PENDAHULUAN

## 1.1 Latar Belakang

Energi masih menjadi pilar utama dalam pembangunan ekonomi dan sosial suatu negara. Di antara berbagai sumber energi, minyak bumi masih memegang peran penting, termasuk di negara berkembang seperti Brasil. Brasil merupakan salah satu produsen minyak terbesar di dunia dengan cadangan minyak terbesar kedua di Amerika Selatan setelah Venezuela. Sejak ditemukannya cadangan minyak besar di wilayah "pre-salt" lepas pantai pada tahun 2006, produksi minyak Brazil meningkat pesat, terutama melalui perusahaan milik negara, Petrobras. Minyak berfungsi sebagai sumber daya inti dalam bidang transportasi, sektor industri, serta produksi listrik. Menurut informasi dari Carbon.org, peran minyak dalam penyediaan energi di Brasil diukur dalam GWh (Gigawatt-hours), yang mengindikasikan tingkat penggunaan atau penghasilannya dalam konteks nasional.

Secara global, output minyak umumnya diungkapkan dalam unit barel atau barel per hari (bpd). Namun, demi analisis antar sektor energi, informasi mengenai minyak dapat diubah ke dalam ukuran energi seperti Gigawatt-jam (GWh). Perubahan ini krusial agar sumbangan minyak dapat lebih mudah dibandingkan dengan sumber energi lainnya, seperti listrik atau biofuel. Sesuai dengan standar konversi, 1 barel minyak setara dengan lebih kurang 0,00177 GWh, dan sebaliknya, 1 GWh setara dengan hampir 564 barel minyak ekuivalen. Apabila data produksi minyak Brasil yang tersedia sudah dalam unit GWh, berarti informasi tersebut telah dikonversi untuk analisis energi baik di tingkat nasional maupun global.

Namun demikian, ketergantungan pada minyak membawa tantangan tersendiri. Variasi harga minyak yang tinggi, ketidakpastian geopolitik, serta tuntutan dunia yang semakin besar untuk beralih ke energi bersih menjadikan perencanaan energi yang berdasarkan minyak perlu dilakukan dengan hati-hati dan berdasar pada data. Dalam hal ini, memperkirakan potensi energi dari minyak di masa depan menjadi langkah krusial bagi pembuatan kebijakan energi nasional, perencanaan investasi, dan strategi peralihan energi.

Untuk melakukan prediksi data energi, pendekatan statistik dan teknologi kecerdasan buatan sering kali dijadikan pilihan. Salah satu metode tradisional yang terbukti efektif dalam pemodelan *time series* adalah ARIMA (Autoregressive Integrated Moving Average), yang dapat mendeteksi pola linier dari data masa lalu. Di sisi lain, kemajuan dalam teknologi pembelajaran mendalam menawarkan pendekatan baru melalui Long Short-Term Memory (LSTM), sebuah model berbasis jaringan saraf yang dapat mengidentifikasi pola non-linier serta hubungan jangka panjang dalam data.

Studi ini bertujuan untuk melakukan perbandingan antara kinerja model ARIMA dan LSTM dalam meramalkan energi (GWh) dari minyak di Brasil. Dengan melakukan analisis ini, diharapkan bisa didapatkan pemahaman mengenai model mana yang paling cocok untuk diterapkan pada perencanaan energi berdasarkan data historis berpola non-linier.

## 1.2 Rumusan Masalah

Berdasarkan latar belakang di atas, berikut merupakan rumusan masalah dari penelitian kali ini:

1. Bagaimana karakteristik pola *time series* dari data energi (GWh) yang berasal dari minyak di Brasil?

2. Sejauh mana model ARIMA bisa digunakan untuk memodelkan serta meramalkan data minyak di Brasil secara tepat?
3. Seberapa efektif model LSTM dalam mengidentifikasi pola non-linear dan meramalkan data minyak Brasil?
4. Model manakah yang menghasilkan prediksi lebih tepat antara ARIMA dan LSTM dalam konteks data minyak di Brasil?

## 1.3 Tujuan

Berdasarkan rumusan masalah, tujuan yang ingin dicapai dalam penelitian ini adalah sebagai berikut:

1. Menganalisis karakteristik dari pola *time series* data energi (GWh) yang berasal dari minyak di Brasil.
2. Membangun serta menilai model ARIMA untuk meramalkan data energi minyak di Brasil dengan memanfaatkan informasi historis yang ada.
3. Membangun dan menilai model LSTM dalam upaya memproyeksikan data energi minyak Brasil serta mengamati bagaimana model ini menangkap pola yang bersifat non-linier.
4. Membandingkan kinerja antara model ARIMA dan LSTM menggunakan metrik evaluasi yang spesifik (RMSE, MAE, dan MAPE) guna mengidentifikasi model mana yang lebih tepat dalam meramalkan energi minyak di Brasil.

## 1.4 Manfaat

Manfaat yang diharapkan pada penelitian kali ini adalah sebagai berikut:

1. Manfaat Akademik, memberikan kontribusi ilmiah dalam pengembangan metode peramalan deret waktu, khususnya dalam membandingkan pendekatan statistik klasik (ARIMA) dan deep learning (LSTM) dalam konteks data energi.
2. Manfaat Praktis, memberikan informasi yang berguna bagi pemangku kebijakan, analis energi, dan sektor industri dalam merancang strategi perencanaan energi berdasarkan hasil prediksi data energi minyak yang lebih akurat.
3. Manfaat Teknis, menyediakan referensi teknis mengenai keunggulan dan keterbatasan model ARIMA dan LSTM, sehingga dapat dijadikan acuan dalam memilih metode prediksi yang sesuai untuk data energi serupa di masa depan.

# BAB 2
# TINJAUAN PUSTAKA

## 2.1 Teori Time Series

Time series adalah sekumpulan data yang dikumpulkan secara berurutan berdasarkan waktu tertentu, seperti harian, mingguan, bulanan, atau tahunan. Data ini memiliki karakteristik khusus, yaitu urutan data sangat penting karena setiap titik data terkait dengan waktu tertentu dan tidak dapat diabaikan. Selain itu, data time series memiliki ketergantungan antar waktu, di mana nilai pada suatu waktu biasanya dipengaruhi oleh nilai sebelumnya. Pola yang sering muncul dalam data time series meliputi:

Trend: Pergerakan data yang menunjukkan kecenderungan naik atau turun dalam jangka panjang.

Musiman (Seasonal): Pola berulang yang terjadi dalam periode tertentu, misalnya bulanan atau kuartalan.

Siklikal: Pola fluktuasi yang terjadi dalam jangka waktu lebih dari satu tahun, biasanya terkait siklus ekonomi.

Analisis time series bertujuan untuk memahami pola-pola tersebut agar dapat digunakan dalam peramalan (forecasting) dan pengambilan keputusan berbasis data. Salah satu aspek penting dalam analisis ini adalah memastikan data bersifat stasioner, yaitu data yang memiliki rata-rata dan variansi konstan sepanjang waktu tanpa pola yang berubah secara sistematis. Data non-stasioner harus diolah terlebih dahulu agar memenuhi asumsi ini sebelum dilakukan pemodelan.

## 2.2 Model ARIMA (AutoRegressive Integrated Moving Average)

Autoregressive Integrated Moving Average (ARIMA) adalah metode yang pertama kali diperkenalkan oleh Box dan Jenkins (1976). Model ARIMA merupakan model yang dapat mengimplementasikan suatu proses deret waktu (time series) yang bersifat tidak stasioner secara univariat (Wei, 2006). Model ARIMA berasal dari model Autoregressive (AR), model Moving Average (MA), dan model campuran. Model campuran terdiri atas model Autoregressive Moving Average (ARMA) yang merupakan kombinasi dari model AR dan MA, serta model Autoregressive Integrated Moving Average (ARIMA) yang merupakan kombinasi dari model AR dan MA dengan unsur differencing. Bentuk umum model ARIMA $(p, d, q)$ seperti pada persamaan (2.1) (Wei, 2006).

$$\phi_p(B)(1-B)^d Y_t = \theta_0 + \theta_q(B)a_t, \qquad (2.1)$$

dengan

$$\phi_p(B) = \left(1 - \phi_1 B - \cdots - \phi_p B^p\right), \qquad (2.2)$$

$$\theta_q(B) = \left(1 - \theta_1 B - \cdots - \theta_q B^q\right), \qquad (2.3)$$

dimana

$\phi p(B)$ : Operator AR
$\theta q(B)$ : Operator MA
$(1 - B)$ : Operator Backward Shift
$Yt$ : Data ke-$t$
$at$ : Residual White Noise ke-$t$
$\theta 0$ : Konstanta

Konstanta ($\theta 0$) memiliki nilai yang berbeda-beda sesuai dengan model ARIMA yang didapatkan. Nilai dari konstanta yang sesuai dengan model ARIMA dijelaskan sebagai berikut.

i. AR(p) : $\theta_0 = (1 - \phi_1 - \cdots - \phi_p)\mu$
ii. MA(q) : $\theta_0 = \mu$
iii. ARMA(p,q) : $\theta_0 = (1 - \phi_1 - \cdots - \phi_p)\mu$

iv. ARIMA(p,d,q) : $\theta_0 = 0$

Prosedur Box-Jenkins berisi tiga tahapan utama untuk membangun model ARIMA (Makridakis, dkk., 1999). Tahapan tersebut dijelaskan sebagai berikut.

**1. Identifikasi Model ARIMA**

Identifikasi model merupakan tahap yang dilakukan untuk mengetahui model ARIMA sementara. Langkah pertama yang harus dilakukan dalam mengidentifikasi model ARIMA adalah data harus stasioner dalam varians dan rata-rata. Salah satu cara untuk mengetahui apakah data sudah stasioner dalam varians atau belum adalah dengan melihat nilai *confidence limit* pada *rounded value* ($\lambda$) di transformasi Box-Cox. Jika nilai *confidence limit* pada *rounded value* sudah melewati 1 maka data sudah stasioner dalam varians. Namun, jika belum, maka data dapat ditransformasi dengan melihat persamaan (2.4) (Wei, 2006).

$$T(Y_t) = Y_t^{(\lambda)} = \begin{cases} \dfrac{Y_t^{\lambda} - 1}{\lambda}, & \lambda \neq 0 \\ \lim\limits_{\lambda \to 0} \dfrac{Y_t^{\lambda} - 1}{\lambda} = ln(Y_t), & \lambda = 0 \end{cases} \quad (2.4)$$

Keterangan:
$Yt$ : Data ke-$t$
$\lambda$ : Parameter transformasi

Jika data sudah stasioner dalam varians, maka dapat dilanjutkan dengan pemeriksaan stasioneritas dalam rata-rata. Pembuktian stasioneritas dalam rata-rata dapat menggunakan uji Augmented Dickey-Fuller (ADF) yang didasarkan pada persamaan (2.5) (Dickey & Fuller, 1979).

$$Yt = \alpha + \delta t + \phi Yt{-}1 + at \quad (2.5)$$

Hipotesis yang digunakan pada uji ADF adalah sebagai berikut.

$H0$ : $\phi = 1$ (Data tidak stasioner dalam rata-rata)

$H1$ : $\phi < 1$ (Data stasioner dalam rata-rata)

Persamaan (2.5) menunjukkan persamaan untuk pengujian ADF dengan tren waktu linear sehingga statistik uji yang digunakan dijelaskan pada persamaan (2.6).

$$\hat{t} = \frac{\hat{\phi} - 1}{\sqrt{Se^2 c}} \quad (2.6)$$

Baris terakhir pada vektor $\boldsymbol{\beta}$ di persamaan (2.7) merupakan nilai dari $\hat{\phi}$.

$$\boldsymbol{\beta} = (\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'\boldsymbol{Y}t \quad (2.7)$$

Diketahui,

$$\boldsymbol{Y}_t' = [Y_2 \quad Y_3 \quad Y_4 \quad \cdots \quad Y_n], \quad (2.8)$$

$$\boldsymbol{X} = \begin{bmatrix} 1 & 1-\frac{n}{2} & Y_1 \\ 1 & 2-\frac{n}{2} & Y_2 \\ 1 & 3-\frac{n}{2} & Y_3 \\ \vdots & \vdots & \vdots \\ 1 & n-1-\frac{n}{2} & Y_{n-1} \end{bmatrix}, \quad (2.9)$$

$$\boldsymbol{\beta}' = [\alpha \quad \delta \quad \phi], \quad (2.10)$$

serta rumus $Se^2$ dan $c$ adalah sebagai berikut.

$$Se^2 = (n-k-1)^{-1}[\boldsymbol{Y}t'(\boldsymbol{I} - \boldsymbol{X}(\boldsymbol{X}'\boldsymbol{X})^{-1}\,\boldsymbol{X}')\boldsymbol{Y}t] \quad (2.11)$$

$$c = (\boldsymbol{X}'\boldsymbol{X})^{-1} \quad (2.12)$$

Data dikatakan stasioner dalam rata-rata jika t-statistik lebih besar dari nilai kritis DF atau p-value kurang dari $\alpha$. Jika data belum stasioner dalam rata-rata, maka perlu dilakukan differencing. Proses differencing orde $d$ ditulis pada persamaan (2.13) (Wei, 2006).

$$\Delta^d Yt = (1-B)^d Yt \quad (2.13)$$

Keterangan:

$Yt$ : Data ke-$t$

$(1-B)$ : Operator Backward Shift

$d$ : Orde differencing

Setelah data sudah stasioner dalam varians dan rata-rata, maka model dapat diidentifikasi menggunakan plot Autocorrelation Function (ACF). ACF adalah suatu representasi dari autoko- relasi antara $Yt$ dan $Yt{-}k$ dari proses yang sama yang hanya terpisah $k$ lag waktu. Plot ACF dapat digambarkan dengan mendapatkan nilai ACF pada persamaan (2.14) (Cryer & Chan, 2008).

$$\hat{\rho}_k = \frac{\sum_{t=k+1}^{n}(Y_t - \bar{Y})(Y_{t-k} - \bar{Y})}{\sum_{t=1}^{n}(Y_t - \bar{Y})^2} \quad (2.14)$$

Selain melihat plot ACF, identifikasi model ARIMA juga dapat dilihat dari plot Partial Autocorrelation Function (PACF). PACF merupakan korelasi antara $Yt$ dan $Yt{-}k$ secara umum akan sama dengan autokorelasi antara $(Yt - \hat{\boldsymbol{Y}}t)$ dan $(Yt{+}k - \hat{\boldsymbol{Y}}t{+}k)$. PACF dapat digunakan untuk mengidentifikasi model ARIMA apakah model tersebut terdapat autoregressive atau tidak. Plot PACF dapat digambarkan dengan menghitung persamaan (2.15) (Cryer & Chan, 2008).

$$\phi_{k,k} = \frac{\hat{\rho}_k - \sum_{j=1}^{k-1}\hat{\phi}_{k-1}\hat{\rho}_{k-j}}{1 - \sum_{j=1}^{k-1}\hat{\phi}_{k-1}\hat{\rho}_{k-j}}, \quad (2.15)$$

$$\text{dengan } \hat{\phi}_{k,j} = \hat{\phi}_{k-1,j} - \hat{\phi}_{k,k}\hat{\phi}_{k-1,k-j},\ j = 1,2,\ldots,k-1.$$

Untuk mendapatkan model ARIMA sementara, struktur plot ACF dan PACF dapat dilihat pada Tabel 2.1 (Wei, 2006).

**Tabel 2. 1** Struktur Plot ACF dan PACF pada Model ARIMA

| Model | ACF | PACF |
|---|---|---|
| AR($p$) | *Dies down* | *Cuf off after lag p* |
| MA($q$) | *Cut off after lag q* | *Dies down* |
| ARMA($p,q$) | *Dies down* | *Dies down* |
| AR($p$) atau MA($q$) | *Cut off after lag q* | *Cut off after lag p* |

## 2. Estimasi Parameter Model ARIMA

Setelah melakukan identifikasi model ARIMA sementara, maka dilanjutkan dengan melakukan estimasi parameter. Estimasi parameter dapat dilakukan menggunakan metode Conditional Least Square dengan cara meminimumkan jumlah kuadrat error. Berikut merupakan contoh estimasi parameter menggunakan model AR (1) (Cryer & Chan, 2008).

$$Yt - \mu = \phi(Yt{-}1 - \mu) + at \qquad (2.16)$$

$$at = (Yt - \mu) - \phi(Yt{-}1 - \mu) \qquad (2.17)$$

Persamaan (2.16) dan (2.17) masing-masing merupakan model dan error dari AR (1). Untuk dapat mengestimasi parameter $\mu$ dan $\phi$ dibutuhkan fungsi *Sum of Square Error* dengan cara menjumlahkan error pada AR (1) dari $t$ = 2 hingga $t$= $n$ sehingga didapatkan persamaan (2.18).

$$S_c(\phi,\mu) = \sum_{t=2}^{n}[(Y_t - \mu) - \phi(Y_{t-1} - \mu)]^2 \qquad (2.18)$$

Estimasi parameter $\mu$ dilakukan dengan cara meminimumkan nilai $Sc(\phi,\mu)$ sebagai berikut.

$$\frac{\partial S_c}{\partial \mu} = \sum_{t=2}^{n} 2[(Y_t - \mu) - \phi(Y_{t-1} - \mu)](\phi - 1) = 0 \qquad (2.19)$$

Dengan mengabaikan nilai $\phi$, estimasi parameter $\mu$ didapatkan pada persamaan (2.20).

$$\hat{\mu} = \frac{1}{(n-1)(1-\hat{\phi})}\left[\sum_{t=2}^{n} Y_t - \hat{\phi}\sum_{t=2}^{n} Y_t\right] = \bar{Y} \qquad (2.20)$$

Kemudian estimasi parameter $\phi$ didapatkan dengan cara yang dituliskan pada persamaan (2.21).

$$\frac{\partial S_c}{\partial \phi} = \sum_{t=2}^{n} 2[(Y_t - \mu) - \phi(Y_{t-1} - \mu)](Y_{t-1} - \mu) = 0 \qquad (2.21)$$

Persamaan (2.22) merupakan hasil dari estimasi parameter $\phi$.

$$\hat{\phi} = \frac{\sum_{t=2}^{n}(Y_t - \bar{Y})(Y_{t-1} - \bar{Y})}{\sum_{t=2}^{n}(Y_{t-1} - \bar{Y})^2} \qquad (2.22)$$

**3. Pengujian Signifikansi Parameter Model ARIMA**

Pengujian signifikansi model ARIMA bertujuan untuk mengetahui apakah parameter yang dihasilkan berpengaruh pada model atau tidak. Model dikatakan baik apabila semua estimasi parameter dalam model signifikan. Berikut merupakan pengujian hipotesis terhadap parameter AR(p) ($\phi p$) atau MA(q) ($\theta q$).

$H0 : \phi_p = 0$ atau $\theta_q = 0$

$H1 : \phi_p \neq 0$ atau $\theta_q \neq 0$

Statistik uji yang digunakan dalam pengujian signifikansi parameter model AR(p) atau MA(q) adalah sebagai berikut.

$$t = \frac{\hat{\phi}_p}{SE(\hat{\phi}_p)} \text{ atau } t = \frac{\hat{\theta}_q}{SE(\hat{\theta}_q)} \qquad (2.23)$$

Parameter model ARIMA dapat dikatakan signifikan apabila nilai $|t| > t_{\frac{\alpha}{2},n-p}$ atau *p-value* < $\alpha$ (Wei, 2006).

**4. Pengujian Kesesuaian Model ARIMA**

Untuk mendapatkan model ARIMA terbaik, terdapat beberapa asumsi residual yang harus dipenuhi, yaitu white noise dan berdistribusi normal. Residual $at$ dikatakan white noise apabila residual merupakan variabel random yang tidak berkorelasi, mempunyai rata-rata sebesar nol dan varians konstan. Pengujian white noise dapat menggunakan uji Ljung-Box dengan hipotesis sebagai berikut.

$H0 : \rho 1 = \rho 2 = \ldots = \rho K = 0$ (Residual white noise)

$H1$ : Minimal terdapat satu $\rho K \neq 0$, $k = 1,2, \ldots, K$ (Residual tidak white noise)

Persamaan (2.24) merupakan statistik uji yang digunakan pada uji Ljung-Box.

$$Q = n(n+2)\sum_{k=1}^{K}(n-k)^{-1}\hat{\rho}_k^2 \qquad (2.24)$$

Dimana $n$ adalah jumlah pengamatan dan $\hat{\boldsymbol{\rho}}k$ adalah ACF residual lag ke-$k$. Apabila nilai $Q > \chi^2(\alpha,k-p-q)$ atau p-value < $\alpha$ maka dapat disimpulkan bahwa residual tidak memenuhi asumsi white noise (Wei, 2006). Asumsi residual yang harus terpenuhi kedua adalah asumsi normalitas. Uji distribusi normal dilakukan menggunakan uji Anderson Darling dengan hipotesis sebagai berikut (Anderson & Darling, 1954).

$H0$ : Residual berdistribusi normal

$H1$ : Residual tidak berdistribusi normal

Statistik uji yang digunakan pada uji Anderson Darling ditulis pada persamaan (2.25).

$$AD = -n - \frac{1}{n}\sum_{i=1}^{n}(2i-1)[ln\,F\,(X_i) + ln(\,1 - F(X_{n-i+1}))] \qquad (2.25)$$

Keterangan:

$n$ : Jumlah ukuran sampel

$F(Xi)$ : Fungsi peluang kumulatif distribusi normal pada sampel ke-i

$F(Xn-i+1)$ : Fungsi peluang kumulatif distribusi normal pada sampel ke-n-i+1

Asumsi residual berdistribusi normal tidak akan terpenuhi jika menghasilkan keputusan tolak $H0$. Keputusan tolak $H0$ terjadi jika $AD >$ critical value atau p-value $< \alpha$.

## 2.3 Pengujian Non Linearitas

Uji nonlinearitas dilakukan untuk mengetahui apakah suatu data mengikuti pola linear atau nonlinear. Salah satu uji nonlinearitas, yaitu uji Terasvirta yang merupakan pengembangan berdasarkan model Neural Network dan termasuk dalam kelompok uji Lagrange Multipler dengan ekspansi Taylor (Terasvirta, Lin, & Granger, 1993). Hipotesis yang digunakan dalam uji Terasvirta adalah sebagai berikut.

$H0$ : $f(x)$ adalah fungsi linear dalam $x$

$H1$ : $f(x)$ adalah fungsi nonlinear dalam $x$

Statistik uji yang digunakan dijelaskan pada persamaan (2.26)

$$\boldsymbol{F = \frac{(SSR_o - SSR)/m}{SSR/(n-r-1-m)}} \qquad (2.26)$$

Keterangan:

$SSRo$ : Jumlah kuadrat error dari regresi $f(x)$ dengan $x$ menghasilkan residual $at$

$SSR$ : Jumlah kuadrat error dari regresi $at$ dengan $x$ dan $m$

$n$ : Jumlah data

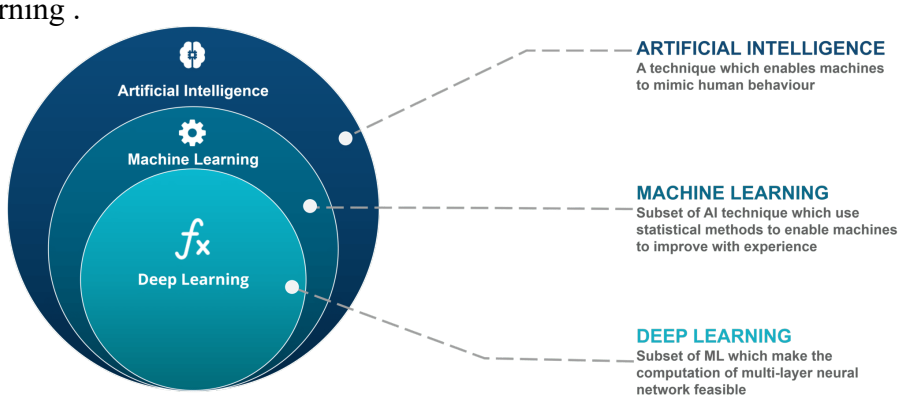
$r$ : Jumlah variabel prediktor awal

$m$ : Jumlah variabel prediktor kuadratik dan kubik.

Model dikatakan nonlinear jika nilai $F > F(m,n-r-1-m)$ atau pvalue $< \alpha$.

## 2.4 *Deep Learning*

Deep Learning merupakan bagian dari kecerdasan buatan yang merupakan bagian dari machine learning. Salah satu metode machine learning yang berdasar pada arsitektur jaringan saraf tiruan otak manusia (atau yang lebih dikenal dan selanjutnya disebut dengan artifical neural network) untuk mengekstrak fitur dari suatu koleksi data yang massif disebut Deep Learning .




**Gambar 2. 1** Penjelasan Tingkatan Deep Learning

Terdapat beberapa contoh arsitektur *Deep Learning* seperti *Deep Neural Networks, Deep Belief Networks, Recurrent Neural Networks, Convolutional Neural Networks,* dan *Long Short Term Memory Network* yang telah diterapkan untuk aplikasi praktis pada bidang-bidang seperti computer vision, pengenalan suara, natural language processing, pengenalan citra, analisis jejaring media sosial, bioinformatika, pembuatan obat, analisis gambar medis, dan finansial dimana telah memberikan hasil yang dapat dibandingkan dan pada beberapa

kasus contoh dapat mengungguli kemampuan manusia yang ahli dibidangnya.

Deep learning terkait erat dengan teori perkembangan otak yang diusulkan oleh para ilmuwan saraf kognitif pada awal 1990 an. Teori-teori perkembangan ini dipakai dalam model komputasi, menjadikannya teori ini pendahulu dari deep learning. Di satu sisi, beberapa varian dari algoritma Backpropagation telah diusulkan untuk meningkatkan realisme pemrosesan. Peneliti lain berpendapat bahwa bentuk pembelajaran mendalam yang tidak diawasi, seperti yang didasarkan pada model generatif hierarkis dan jaringan kepercayaan yang mendalam, mungkin lebih dekat dengan realitas biologis. Dalam hal ini, model jaringan saraf generatif telah dikaitkan dengan bukti neurobiologis tentang pemrosesan berbasis sampel di korteks serebral.

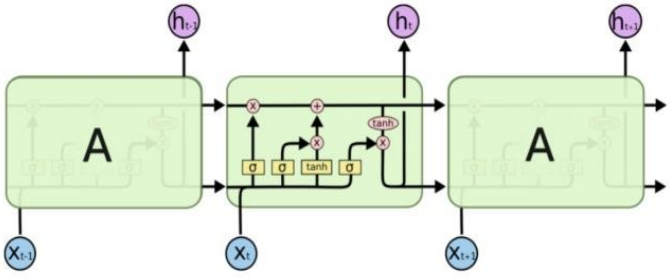
## 2.5 Jaringan Syaraf Tiruan/*Neural Network* (NN)

Menurut Fausett (1944), jaringan syaraf tiruan (JST) atau neural network (NN) adalah sistem pemrosesan informasi yang dimana karakteristik kemampuan yang mirip dengan jaringan syaraf manusia. JST telah dikembangkan sebagai generelalisasi secara matematis dari jaringan syaraf manusia, dengan asumsi sebagai berikut.

1. Pemrosesan informasi terjadi pada banyak elemen sederhana yang disebut neuron.
2. Sinyal dikirimkan antar neuron melalui suatu koneksi penghubung.
3. Setiap koneksi penghubung antar neuron mempunyai bobot yang akan memperkuat atau memperlemah sinyal (bobot yang bernilai positif akan memperkuat sinyal, dan sebaliknya).
4. Setiap neuron menerapkan sebuat fungsi aktivasi (yang biasanya tidak linier) ke setiap jaringan input (jumlah terboboti dari sinyal input) untuk menentukan sinyal outputnya.

## 2.6 Model LSTM (Long Short-Term Memory)

Long Short-Term Memory (LSTM) merupakan pengem- bangan dari Recurrent Neural Network (RNN) yang diperkenalkan oleh Sepp Hochreiter & Jurgen Schmidhuber (1997). LSTM me- nyimpan informasi mengenai memori yang ada di masa lalu dalam jangka pendek maupun jangka panjang. LSTM digunakan dalam berbagai bidang, salah satunya data time series (Aldi, Jondri, & Aditsania, 2018). LSTM terdiri dari banyak cell state. Cell state berfungsi untuk menghubungkan semua output layer yang ada pada LSTM. LSTM dapat menambah dan/atau menghapus infor- masi dari cell state. Pada masing-masing cell state tersebut terdapat empat proses fungsi aktivasi yang selanjutnya disebut gates. Gates berfungsi untuk mengatur apakah informasi akan diteruskan atau diberhentikan. Empat gates pada masing-masing cell state tersebut adalah forget gates, input gates, new cell state candidates, dan out- put gates yang arsitekturnya dapat dilihat pada Gambar.....



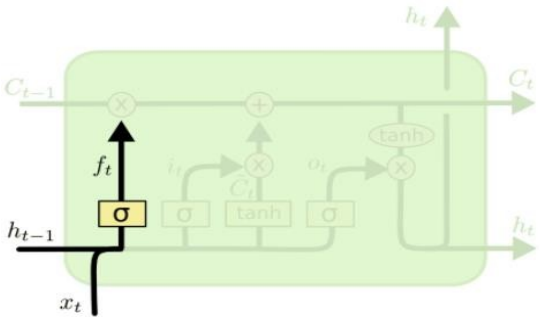

**Gambar 2. 2** Arsitektur LSTM

Langkah pertama yang dilakukan pada LSTM adalah memu- tuskan informasi yang akan dibuang dari cell state. Keputusan ini dibuat oleh lapisan sigmoid yang disebut forget

gate. Forget gates akan memproses $ht-1$ dan $xt$ sebagai input sehingga menghasilkan output berupa angka antara 0 sampai 1 pada cell state. Forget gate dijelaskan pada persamaan (2.27).

$$ft = \sigma(W_f \text{ x } [h_{t-1}, x_t] + b_f) \qquad (2.27).$$

Keterangan
$ft$ : Forget gate
$\sigma$ : Fungsi sigmoid
$Wf$ : Nilai weight untuk forget gate
$ht-1$ : Nilai output ke-$t-1$
$xt$ : Nilai input ke-$t$
$bf$ : Nilai bias pada forget gate

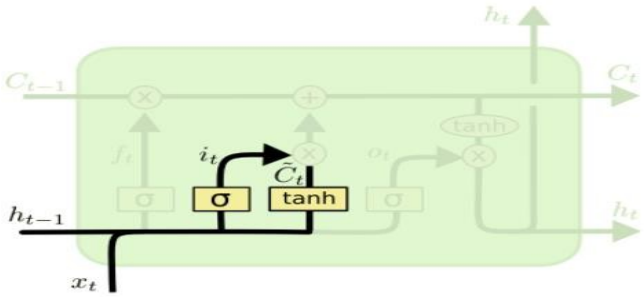



**Gambar 2. 3** Forget Gates LSTM

Langkah kedua, yaitu memutuskan informasi apa yang akan disimpan pada cell state. Pada langkah ini terdapat dua bagian. Ba- gian pertama, yaitu input gate, berfungsi untuk memutuskan nilai mana yang akan diperbaharui. Bagian kedua, yaitu $tanh$ layer, ber- fungsi untuk membuat satu kandidat dengan nilai baru yang dapat ditambahkan ke dalam cell state. Kemudian output dari input gate dan $tanh$ layer digabungkan untuk memperbaharui cell state. Input gate dan $tanh$ layer dijelaskan pada persamaan (2.28) dan (2.29).

$$it = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \qquad (2.28)$$
$$\tilde{C}_t = tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \qquad (2.29)$$

Keterangan
$it$ : Input gate
$\sigma$ : Fungsi sigmoid
$Wi$ : Nilai weight untuk input gate
$ht-1$ : Nilai output ke-$t-1$
$xt$ : Nilai input ke-$t$
$bi$ : Nilai bias pada input gate
$\tilde{C}t$ : Nilai baru yang dapat ditambahkan ke cell state
$WC$ : Nilai weight untuk $tanh$ layer
b$C$ : Nilai bias pada $tanh$ layer

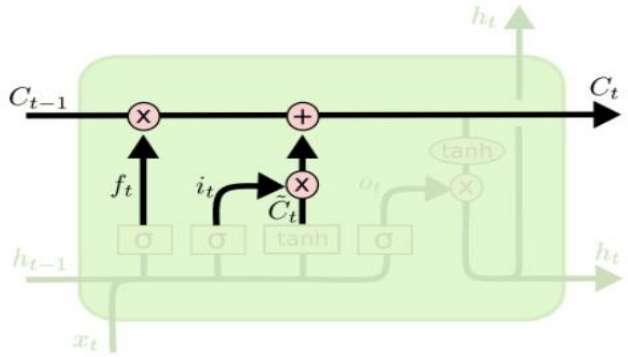



**Gambar 2. 4** Input Gate dan tanh Layer pada LSTM

Langkah ketiga pada LSTM adalah memperbaharui cell state yang lama dengan cell state yang baru. Pembaharuan cell state ini dijabarkan pada persamaan berikut.

$$Ct = ft * Ct{-}1 + it * \tilde{C}t \qquad (2.30)$$

Keterangan
$C_{t-1}$ : Nilai yang telah diperbaharui ke-$t-1$
$C_t$ : Nilai yang telah diperbaharui ke-$t$



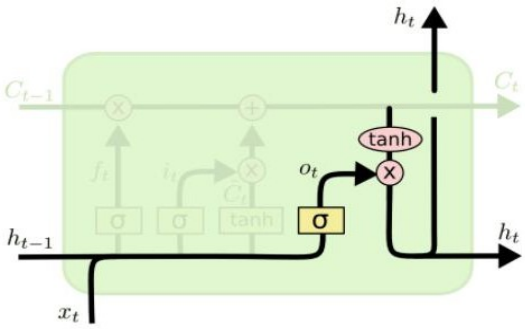

**Gambar 2. 5** Pembaharuan Cell pada LSTM

Langkah terakhir yang harus dilakukan, yaitu memasukkan output dari cell state ke dalam $tanh$ layer untuk mengubah nilai menjadi dalam rentang -1 hingga 1 serta mengalikan angka tersebut dengan sigmoid gate, agar output yang dihasilkan sesuai dengan keputusan sebelumnya. Output gate dijabarkan pada persamaan (2.31) kemudian dapat diuraikan menjadi persamaan (2.32).

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \qquad (2.31)$$

$$h_t = o_t * tanh(C_t) \qquad (2.32)$$

Keterangan
$Wo$ : Nilai weight untuk output gate
$bo$ : Nilai bias pada output gate
$ot$ : Output gate
ht : Nilai output ke-$t$




**Gambar 2. 6** Output Gate pada LSTM

## 2.7 Evaluasi Model Time Series

Setelah melakukan prediksi, diperlukan evaluasi untuk mengetahui tingkat kesalahan hasil prediksi. Evaluasi yang pertama adalah menggunakan Mean Absolute Error (MAE). Mean Absolute Error (MAE) cocok digunakan jika outlier adalah bagian dari data yang tidak valid. MAE tidak terlalu menilai outlier dalam data training, sehingga memberikan ukuran kinerja yang stabil dan terbatas untuk model. Sebaliknya, jika set pengujian juga mengandung banyak outlier, kinerja model tidak akan optimal. Persamaan dari MAE dapat dituliskan sebagai berikut:

$$\boldsymbol{MAE} = \frac{\mathbf{1}}{\boldsymbol{n}}\sum_{\boldsymbol{i=1}}^{\boldsymbol{n}}|\widehat{\boldsymbol{\epsilon_t}} - \boldsymbol{\epsilon_t}| \qquad (2.33)$$

Metode evaluasi kedua yang digunakan adalah Mean Absolute Percentage Error (MAPE). Mean Absolute Percentage Error (MAPE) adalah metrik evaluasi yang mengukur tingkat kesalahan relatif dalam peramalan. MAPE menghitung kesalahan absolut pada setiap periode, kemudian membaginya dengan nilai observasi yang sebenarnya untuk periode tersebut, dan akhirnya menghitung rata-rata kesalahan persentase absolut tersebut. Pendekatan ini berguna terutama ketika skala atau besarnya variabel yang diestimasi memiliki signifikansi penting dalam menilai akurasi peramalan dibandingkan dengan nilai sebenarnya. Persamaannya dapat dirumuskan sebagai berikut:

$$MAPE = \left(\frac{1}{n}\sum_{i-1}^{t}\left|\frac{\sigma_t^2 - \widehat{\sigma_t^2}}{\sigma_t^2}\right|\right) \times 100\% \qquad (2.34)$$

Metode evaluasi ketiga yang digunakan adalah Root Mean Square Error (RMSE). RMSE (Root Mean Square Error) adalah ukuran statistik yang digunakan untuk mengevaluasi perbedaan antara nilai yang diprediksi dan nilai yang diamati. Secara sederhana, RMSE adalah akar kuadrat dari rata-rata kesalahan kuadrat. Rumus RMSE adalah:

$$RMSE = \left(\frac{\sum(y_i - \hat{y}_i}{n}\right)^{1/2} \qquad (2.35)$$

Semakin kecil nilai MAE, RMSE, dan MAPE, maka model dianggap semakin akurat. Evaluasi juga dapat membandingkan performa beberapa model, misalnya ARIMA dan LSTM, pada data yang sama. Penelitian menunjukkan bahwa LSTM seringkali memberikan hasil yang lebih baik pada data dengan pola kompleks dan non-linear, sementara ARIMA

cocok untuk data yang lebih sederhana dan stasioner. Selain itu, validasi silang (cross-validation) dan uji kestasioneran data juga penting dalam proses evaluasi model time series.

## 2.8 *Oil Production* Brasil

Produksi minyak di Brasil menunjukkan tren peningkatan yang signifikan dalam beberapa tahun terakhir. Pada tahun 2023, produksi minyak Brasil mencapai 3,502 juta barel per hari (bbl/d), meningkat sebesar 12,52% dibandingkan tahun sebelumnya yang sebesar 3,112 juta bbl/d. Pertumbuhan ini menandai rata-rata peningkatan tahunan sekitar 6,71% sejak beberapa tahun terakhir. Pada bulan Mei 2024, produksi minyak Brasil tercatat sebesar 3,318 juta bbl/d, naik 3,9% dibandingkan bulan sebelumnya dan 3,6% dibandingkan Mei 2023. Selain itu, produksi gas alam juga meningkat, mencapai 145,63 juta meter kubik per hari, naik 6,6% dari April 2024 dan 0,8% dari Mei 2023.

Sebagian besar produksi minyak dan gas ini berasal dari wilayah pre-salt, yang pada Mei 2024 menghasilkan total 3,314 juta barel setara minyak per hari (boe/d), atau sekitar 78,3% dari total produksi nasional. Produksi dari 145 sumur di wilayah ini mencapai 2,599 juta bbl/d minyak dan 113,73 juta $m^3$/d gas alam. Produksi minyak Brasil didominasi oleh ladang lepas pantai, yang menyumbang 97,5% dari total produksi minyak dan 86,2% dari produksi gas alam. Perusahaan Petrobras, baik secara mandiri maupun dalam konsorsium, mengelola sekitar 88,88% dari total produksi minyak dan gas di negara ini.

Melihat ke depan, Brasil berencana untuk terus meningkatkan produksi minyaknya. Menurut Institut Minyak Brasil (IBP), produksi minyak diperkirakan akan mencapai 3,6 juta bbl/d pada tahun 2025, meningkat sekitar 6% dari rata-rata produksi tahun 2024 yang sekitar 3,4 juta bbl/d. Proyeksi ini menunjukkan langkah strategis Brasil untuk memperkuat posisinya sebagai kekuatan energi, khususnya dalam sumber energi tradisional.

Meskipun proyeksi awal dari Rencana Ekspansi Energi Sepuluh Tahun memperkirakan peningkatan produksi hingga lebih dari 4 juta bbl/d (sekitar 10% kenaikan), IBP memberikan perkiraan yang lebih konservatif berdasarkan data dari berbagai sumber terpercaya seperti S&P Global, Badan Nasional Minyak, Gas Alam dan Biofuel Brasil (ANP), serta OPEC. Petrobras diperkirakan akan menjadi pendorong utama pertumbuhan produksi ini.

Secara keseluruhan, produksi minyak Brasil menunjukkan peningkatan yang stabil dan berkelanjutan, didukung oleh pengembangan ladang minyak pre-salt dan investasi dari perusahaan-perusahaan utama di sektor ini. Hal ini menegaskan peran penting Brasil sebagai salah satu produsen minyak utama di Amerika Selatan dengan potensi pertumbuhan yang kuat di masa depan.

## 2.9 Google Colab

Google Colab adalah lingkungan pengembangan terintegrasi (IDE) berbasis cloud yang memungkinkan pengguna untuk menulis, menjalankan, dan berbagi kode Python secara interaktif melalui browser web. Colab merupakan versi khusus dari Jupyter Notebook yang berjalan di infrastruktur Google Cloud, sehingga pengguna tidak perlu menginstal perangkat lunak secara lokal dan dapat mengakses proyek dari mana saja dengan koneksi internet. Beberapa manfaat utama Google Colab antara lain:

a) Eksekusi berbasis cloud: Memanfaatkan sumber daya komputasi Google Cloud, termasuk akses gratis ke GPU dan TPU, sehingga cocok untuk proyek machine learning dan analisis data yang membutuhkan daya komputasi tinggi.
b) Kolaborasi real-time: Memungkinkan beberapa pengguna bekerja bersama dalam satu notebook secara simultan, memudahkan berbagi dan pengembangan proyek secara kolaboratif.
c) Mudah digunakan dan terintegrasi: Sudah tersedia banyak pustaka Python populer yang siap pakai, serta integrasi mudah dengan Google Drive dan GitHub untuk

manajemen file dan versi kode.

d) Fitur lanjutan: Mendukung perintah terminal langsung, instalasi pustaka tambahan, visualisasi data secara interaktif, dan penyimpanan otomatis notebook dalam format .ipynb atau .py.

## 2.10 R-Studio

R-Studio adalah lingkungan pengembangan terintegrasi (IDE) open-source yang dirancang khusus untuk bahasa pemrograman R, yang merupakan bahasa populer untuk komputasi statistik dan visualisasi data. R-Studio menyediakan berbagai fitur produktivitas yang memudahkan pengguna dalam melakukan analisis data, seperti penyorotan sintaks, penyelesaian kode otomatis, dan indentasi cerdas. Beberapa keunggulan R-Studio dalam analisis data adalah:

a) Workflow yang efisien: Menyediakan alat untuk manipulasi data, visualisasi, dan pelaporan dalam satu platform terpadu, sehingga mempercepat proses analisis dan interpretasi data.

b) Dukungan komunitas dan ekosistem luas: R dan R-Studio didukung oleh komunitas yang aktif serta banyak paket tambahan yang memperluas kemampuan analisis statistik dan grafis.

c) Fitur manipulasi data: Memungkinkan pembersihan, transformasi, dan pengelompokan data secara mudah untuk persiapan analisis lebih lanjut.

d) Visualisasi data yang kuat: Mendukung pembuatan grafik dan visualisasi interaktif yang membantu mengungkap pola dan tren dalam data.

# BAB 3
# METODOLOGI PENELITIAN

## 3.1 Sumber Data

Data yang digunakan dalam penelitian ini adalah data sekunder yang diambil dari webiste Carbon.org terkait dengan minyak Brasil (GWh) dalam hari dari tahun 2024 sampai 2025 dengan jumlah keseluruhan data adalah 528 yang dapat diakses secara terbuka pada https://power.carbonmonitor.org/org.

## 3.2 Variabel Penelitian

Variabel yang digunakan pada penelitian ini dijelaskan pada table 3.1.

**Tabel 3. 1** Variabel Penelitian

| No | Nama Variabel | Tipe Data | Deskripsi | Peran dalam Model |
|---|---|---|---|---|
| 1 | date | Time series | Tanggal observasi data energi (format dd/mm/yyyy) | Indeks waktu |
| 2 | country | Kategorikal | Negara asal data, yaitu *Brasil* | Informasi pendukung |
| 3 | sector | Kategorikal | Jenis sektor energi, yaitu *Oil* | Informasi pendukung |
| 4 | GWh | Numerik kontinu | Jumlah energi minyak dalam satuan Gigawatt-hours (GWh) per hari | Variabel target (Y) |

Tabel di atas menunjukkan variabel-variabel yang digunakan dalam penelitian ini. Variabel utama yang menjadi fokus adalah jumlah energi harian dari minyak (GWh) yang berperan sebagai variabel target dalam proses peramalan. Di sisi lain, variabel tanggal (date) berfungsi sebagai penanda waktu dalam model deret waktu. Variabel country dan sector hanya berperan sebagai informasi identifikasi karena semua data berasal dari sektor minyak di Brasil. Data ini akan digunakan lebih lanjut dalam pengembangan model dengan menerapkan metode ARIMA dan LSTM untuk memprediksi konsumsi energi minyak di masa depan. Di sisi lain, variabel tanggal (date) berfungsi sebagai penanda waktu dalam model deret waktu. Variabel country dan sector hanya berperan sebagai informasi identifikasi karena semua data berasal dari sektor minyak di Brasil. Data ini akan digunakan lebih lanjut dalam pengembangan model dengan menerapkan metode ARIMA dan LSTM untuk memprediksi konsumsi energi minyak di masa depan.

## 3.3 Tahapan Analisis

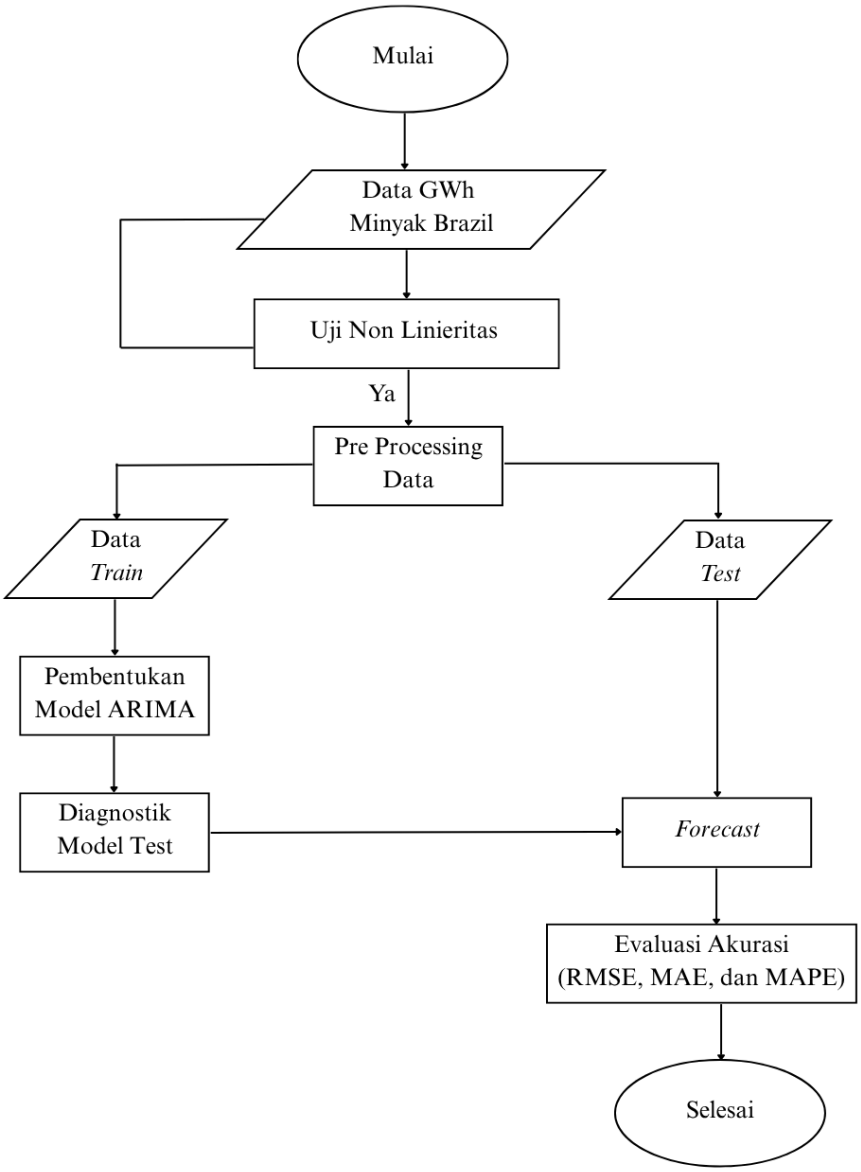
Langkah-langkah yang dilakukan dalam penelitian ini adalah sebagai berikut.

1. Melakukan studi literatur terkait dengan peramalan deret waktu, khususnya metode ARIMA dan Long Short-Term Memory (LSTM)
2. Mengumpulkan data konsumsi energi dari minyak (GWh) harian di Brasil dari situs Carbon.org dalam periode 1 Januari 2024 hingga 12 Juni 2025.
3. Mendeskripsikan karakteristik data konsumsi energi dari minya (GWh) harian di Brasil.
4. Melakukan pengujian nonlinearitas.

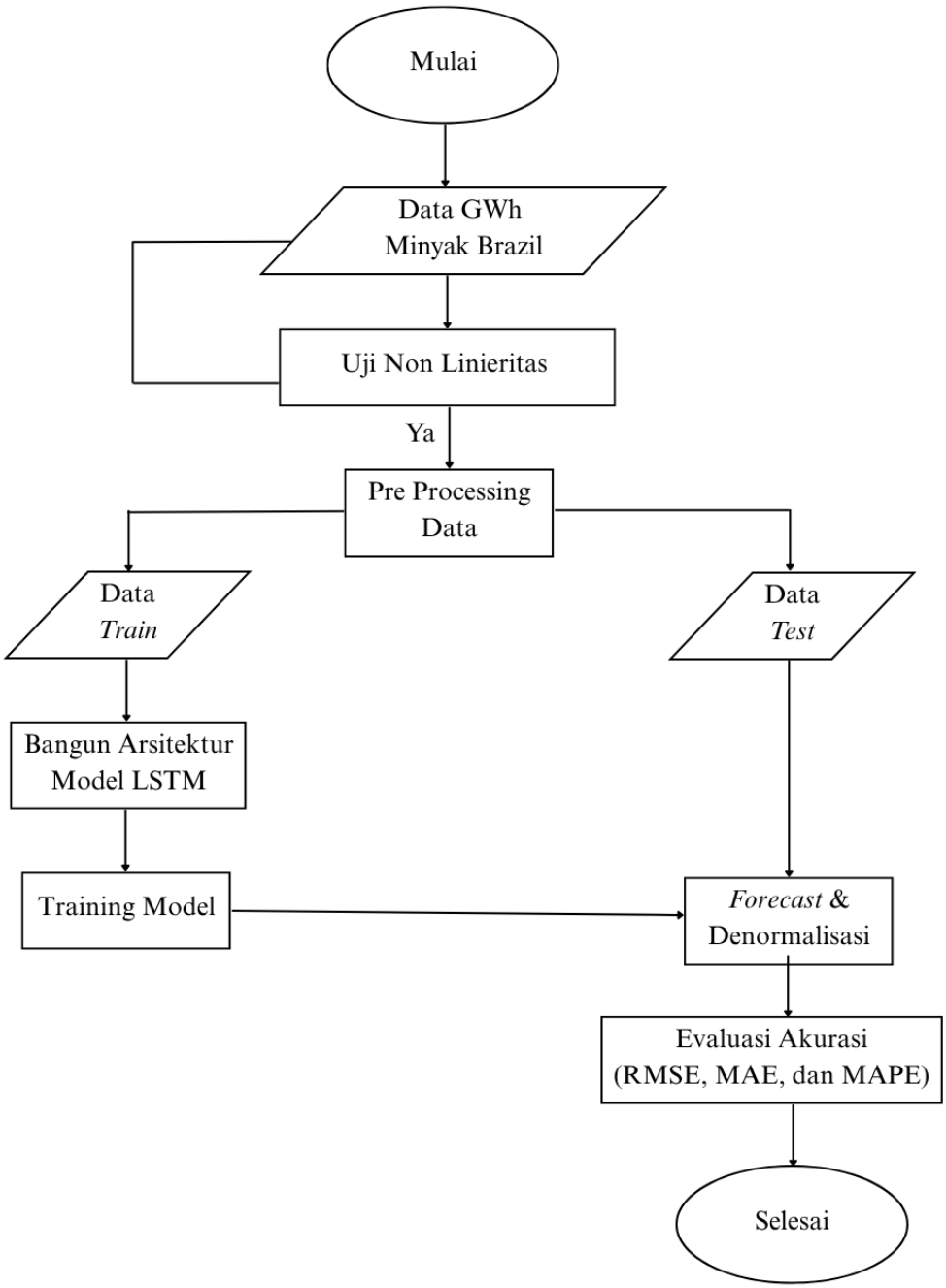
5. Melakukan pemodelan dan peramalan menggunakan metode ARIMA. Langkah-langkah dalam metode ARIMA adalah sebagai berikut:
    a. Memisahkan data menjadi data latih (*in-sample*) dan data uji (*out-sample*).
    b. Melakukan uji stasioneritas dalam mean dan varians terhadap data.
    c. Melakukan identifikasi orde ARIMA menggunakan plot ACF dan PACF.
    d. Melakukan estimasi parameter model ARIMA.
    e. Melaksanakan pengujian diagnostik untuk menilai kesesuaian model.
    f. Melakukan peramalan menggunakan model ARIMA yang telah dibuat.
    g. Menghitung nilai RMSE, MAE, dan MAPE untuk menilai kinerja model.
6. Melakukan pemodelan dan prediksi menggunakan teknik Long Short-Term Memory (LSTM). Rincian langkah-langkah dalam teknik LSTM adalah sebagai berikut:
    a. Melaksanakan normalisasi pada data numerik.
    b. Memisahkan data kedalam set pelatihan (*in-sample*) dan set pengujian (*out-sample*).
    c. Mengatur panjang jendela (*time steps*) untuk input LSTM.
    d. Mengubah susunan *input* menjadi format tiga dimensi (*batch, time steps, feature*).
    e. Menentukan desain jaringan (jumlah *neuron* dan *hidden layer*).
    f. Melatih model LSTM menggunakan data *training*.
    g. Melakukan denormalisasi data.
    h. Menghitung nilai RMSE, MAE, dan MAPE untuk menilai kinerja prediksi.
7. Membandingkan kinerja model ARIMA dan LSTM berdasar nilai evaluasi (RMSE, MAE, dan MAPE) pada set pengujian.
8. Menyusun kesimpulan dan memberikan saran berdasarkan temuan serta perbandingan model yang telah dilakukan.

## 3.4 Diagram Alir

Langkah analisis pada penelitian ini dapat digambarkan dalam Gambar 3.1 dan Gambar 3.2.




**Gambar 3. 1** Diagram Alir ARIMA

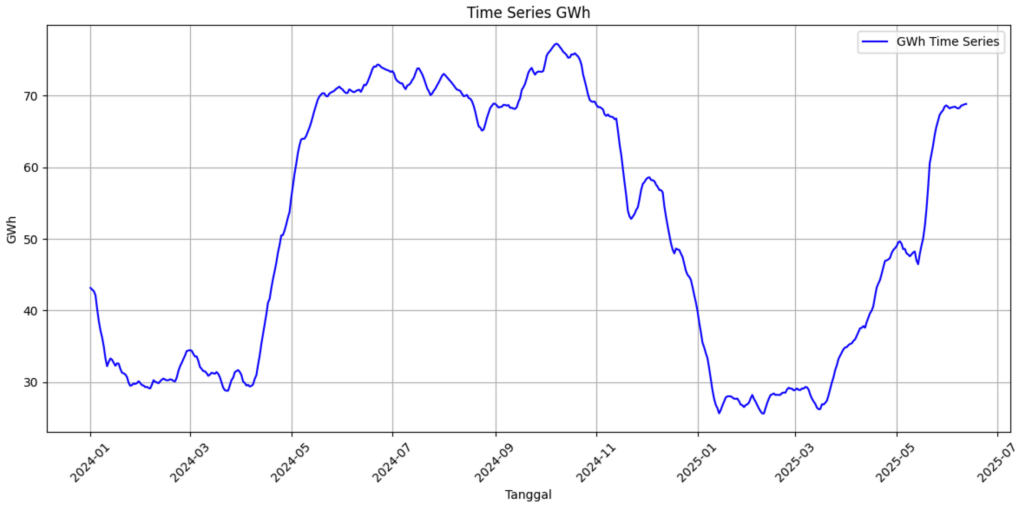



**Gambar 3. 2** Diagram Alir LSTM

# BAB 4
# HASIL DAN PEMBAHASAN

## 4.1 Karakteristik Data GWh Minyak di Brasil

Sebelum dilakukan proses analisis lebih lanjut, terlebih dahulu dilakukan eksplorasi awal terhadap data guna memperoleh gambaran umum pola dan karakteristik dari data GWh minyak di Brasil. Eksplorasi dilakukan dengan menyajikan plot data deret waktu untuk melihat tren, fluktuasi musiman, serta adanya kemungkinan anomali pada data. Visualisasi data deret waktu GWh sektor minyak di Brazil ditampilkan pada Gambar 4.1.




**Gambar 4. 1** Plot Data Time Series

Pada Gambar 4.1, dapat dilihat bahwa penggunaan energi (dalam GWh) di sektor minyak Brasil menunjukkan variasi yang cukup berarti sepanjang waktu yang diamati. Tidak terdapat pola musiman yang tetap, namun ada beberapa perubahan tajam yang terlihat jelas. Di awal tahun 2024, tingkat konsumsi energi berada pada posisi yang cukup rendah, kemudian meningkat pesat sampai mencapai puncaknya sekitar bulan Mei hingga Juni 2024. Setelah periode tersebut, terjadi penurunan yang signifikan yang berlanjut hingga awal tahun 2025. Penurunan ini kemudian diikuti oleh fase pemulihan yang relatif cepat, dengan tren yang mulai meningkat kembali pada pertengahan tahun 2025. Fluktuasi yang cukup ekstrim ini menunjukkan kemungkinan adanya faktor eksternal yang memengaruhi data, seperti perubahan kebijakan, gangguan pasokan, atau dinamika pasar energi global. Pola yang tidak stabil ini juga mungkin mencerminkan adanya anomali atau perubahan struktural dalam sektor minyak di Brasil selama periode waktu tersebut.

## 4.2 Uji Non Linieritas Data

Dalam studi ini, akan ada analisis perbandingan kinerja antara model ARIMA (linier) dan LSTM (non-linier) dalam memprediksi data konsumsi energi (GWh) di sektor minyak Brasil. Dengan demikian, sangat penting untuk terlebih dahulu memverifikasi apakah data yang digunakan memiliki elemen non-linier. Uji dilakukan dengan menggunakan Teraesvirta Neural Network Test, yang bertujuan untuk menemukan adanya struktur non-linier dalam data deret waktu. Hasil dari uji tersebut ditunjukkan pada tabel 4.1 berikut.

**Tabel 4. 1** Hasil Uji Non Linieritas (Teraesvirta Neural Network Test)

| Statistik Uji $\chi^2$ | Derajat Bebas (df) | p-value |
|---|---|---|
| 6.3201 | 2 | 0.04242 |

Berdasarkan hasil uji, diperoleh p-value sebesar 0.04242 yang lebih kecil dari tingkat signifikansi 5%. Dengan demikian, hipotesis nol (bahwa data bersifat linier) ditolak. Artinya, data memiliki karakteristik non-linier, sehingga model non-linier seperti LSTM layak digunakan dan dibandingkan dengan model linier seperti ARIMA dalam penelitian ini.

## 4.3 Hasil Preprocessing ARIMA

a) Pengecekan Missing Value
   Pada data produksi minyak Brasil, dilakukan pengecekan nilai yang hilang (missing values) pada kolom GWh. Dari hasil pengecekan ditemukan tidak ada nilai yang hilang, sehingga tidak diperlukan penghapusan data. Hal ini memastikan data yang digunakan lengkap dan valid.
b) Pembentukan Objek Time Series
   Data produksi minyak yang sudah bersih dibentuk menjadi objek time series dengan frekuensi 365, yang merepresentasikan data harian. Pembentukan objek time series ini penting untuk analisis deret waktu dan pemodelan selanjutnya.
c) Pembagian Data Training dan Testing
   Data yang sudah stasioner kemudian dibagi menjadi dua bagian, yaitu data training sebanyak 83% dan data testing sebanyak 17%. Data training digunakan untuk membangun model, sedangkan data testing digunakan untuk menguji performa model yang telah dibangun.
d) Pengujian Stasioneritas Data Asli
   Pengujian stasioneritas dilakukan menggunakan uji Augmented Dickey-Fuller (ADF). Didapatkan hasilnya :

**Tabel 4. 2** Hasil Uji Stationer ADF Data Asli

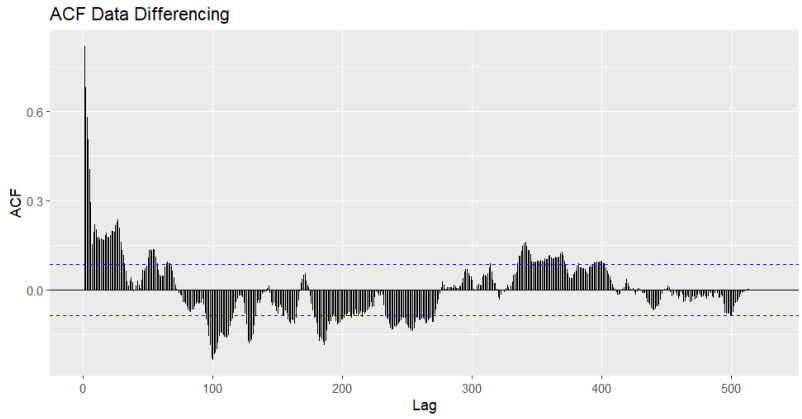
| ADF p-value (Original) |
|---|
| 0.6957036 |

   Hasil uji menunjukkan nilai p-value sebesar 0,6957, yang lebih besar dari 0,05, sehingga data asli dinyatakan tidak stasioner. Kondisi ini mengindikasikan adanya pola trend atau ketergantungan waktu yang harus diatasi agar model dapat bekerja dengan baik.
e) Transformasi Differencing
   Untuk mengatasi ketidakstasioneran, dilakukan differencing orde pertama pada data.

**Tabel 4. 3** Hasil Uji Stationer Setelah Transformasi

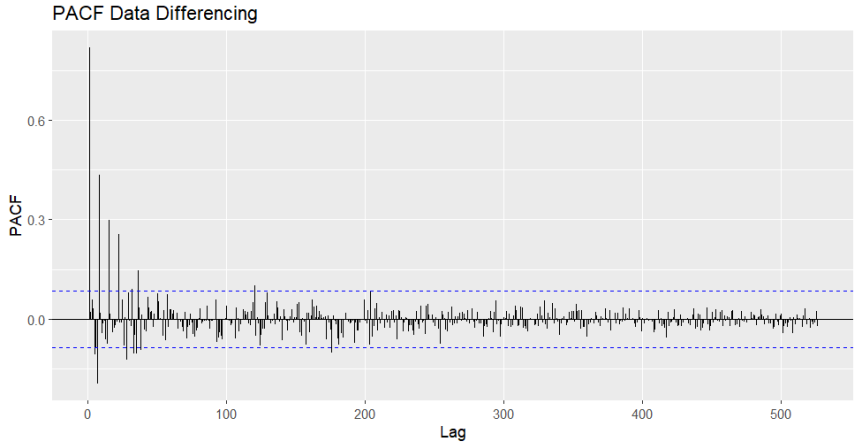
| ADF p-value (Differencing) |
|---|
| 0.01 |

   Setelah differencing, dilakukan kembali uji ADF yang menghasilkan nilai p-value sebesar 0,01 (kurang dari 0,05), menandakan data sudah menjadi stasioner. Visualisasi data hasil differencing juga dilakukan untuk mengamati pola dan korelasi data melalui plot time series, Autocorrelation Function (ACF), dan Partial Autocorrelation Function (PACF).




**Gambar 4. 2** Gambar ACF data Oil Production Brazil

ACF di atas menunjukkan nilai autokorelasi data hasil differencing pada berbagai lag. Terlihat bahwa pada lag awal, nilai autokorelasi cukup tinggi dan signifikan, namun secara umum nilai autokorelasi menurun dan sebagian besar berada di dalam batas signifikansi (ditandai garis biru putus-putus). Hal ini mengindikasikan bahwa setelah dilakukan differencing, sebagian besar autokorelasi sudah hilang, yang merupakan indikasi data telah menjadi lebih stasioner.




**Gambar 4. 3** Gambar PACF data Oil Production Brazil

PACF menunjukkan nilai partial autokorelasi pada berbagai lag setelah differencing. Terlihat bahwa hanya beberapa lag awal yang memiliki nilai signifikan, sedangkan lag-lag berikutnya cenderung berada di sekitar nol dan di dalam batas signifikansi. Pola ini menunjukkan bahwa setelah differencing, pengaruh lag terhadap data sudah sangat berkurang, yang memperkuat indikasi bahwa data telah stasioner. Pola ACF yang menurun secara bertahap dan PACF yang signifikan pada beberapa lag awal dapat digunakan sebagai dasar dalam menentukan parameter orde AR (p) dan MA (q) pada model ARIMA.

## 4.4 Hasil dan Analisis Model ARIMA

Model ARIMA dengan konfigurasi parameter (3,1,4) telah diterapkan untuk data pelatihan menggunakan transformasi Box-Cox dengan nilai λ sebesar 1. Hasil estimasi model menunjukkan nilai-nilai koefisien autoregressive (AR) dan moving average (MA) sebagai berikut:

**Tabel 4. 4** Model ARIMA

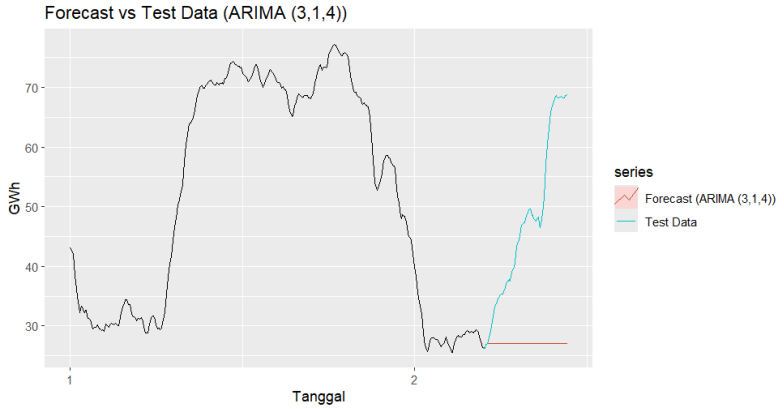
| Model | Koefisien | Error (s.e) |
|---|---|---|
| AR(1) | 0,3440 | 0.0513 |
| AR(2) | 0,4471 | 0,0488 |
| AR(3) | -0,3477 | 0,0518 |
| MA(1) | 0,5485 | 0,0290 |
| MA (2) | -0,2647 | 0,0280 |
| MA (3) | 0,4790 | 0,0401 |
| MA (4) | 0,8923 | 0,0347 |

Model memiliki variansi residual ($\sigma^2$) sebesar 0.1074, dengan nilai log-likelihood sebesar -132.63. Kriteria evaluasi model pada data pelatihan menunjukkan hasil AIC: 281.26; AICc: 281.59; BIC: 313.9. Evaluasi terhadap error data pelatihan memberikan hasil:

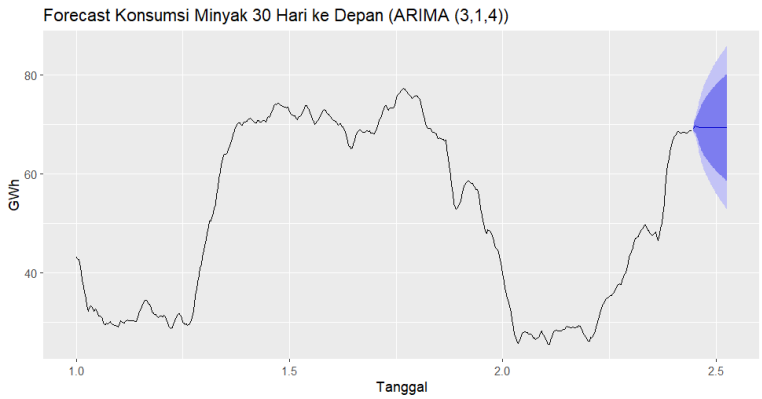
**Tabel 4. 5** Evaluasi Model ARIMA

| Metrik Evaluasi | Data *Train* | Data *Test* |
|---|---|---|
| MAE | 0,2378754 | 20,4873 |
| RMSE | 0.3247296 | 24.4954 |
| MAPE | 54.21% | 38.26% |

Nilai ACF1 residual sebesar -0.0107 menunjukkan bahwa residual dari model ini relatif tidak berkorelasi, yang menandakan model telah menangkap pola data secara efektif. Dengan model ARIMA (3,1,4) ini didapatkan forecastingnya adalah :




**Gambar 4. 4** Gambar Perbandingan Forecast dan Data Test




**Gambar 4. 5** ***Gambar Hasil Forecast 30 Hari Kedepan***

Berdasarkan hasil visualisasi, model ARIMA (3,1,4) memiliki keterbatasan dalam menangkap pola musiman atau fluktuasi mendadak pada data aktual. Hal ini dapat menyebabkan peningkatan nilai error, seperti yang ditunjukkan pada nilai MAPE (38.26%) dan RMSE (24.4954). Untuk meningkatkan akurasi, dapat dipertimbangkan penggunaan pendekatan model lain seperti model non-linear.

## 4.5 Hasil dan Analisis Model LSTM

Dalam studi ini, pendekatan yang diterapkan untuk memprediksi konsumsi energi di sektor minyak Brazil adalah model Long Short-Term Memory (LSTM), yang merupakan bagian dari Recurrent Neural Network (RNN). LSTM terkenal efektif dalam mengatasi masalah yang berkaitan dengan data waktu karena kemampuannya dalam menyimpan informasi jangka panjang dan menghindari isu vanishing gradient yang sering terjadi pada RNN konvensional. Dataset yang dianalisis mencakup data harian terkait konsumsi energi sektor minyak (dalam satuan GWh) dari 01 Januari 2024 sampai 12 Juni 2025. Jumlah keseluruhan data yang digunakan adalah 528 pengamatan. Sasaran dari pemodelan ini adalah untuk memproyeksikan konsumsi energi selama 30 hari ke depan berdasarkan pola konsumsi yang ada sebelumnya.

a) Pre-Processing Data

Tahapan awal dalam pembentukan model dimulai dari proses pra-pemrosesan data. Pertama, kolom tanggal ("date") diubah ke format datetime agar data dapat disusun secara kronologis dan dianalisis secara runtut berdasarkan waktu. Setelah itu, data diurutkan berdasarkan tanggal. Variabel target yang diprediksi adalah kolom "GWh" yang menunjukkan jumlah konsumsi energi. Nilai-nilai pada kolom ini diubah ke dalam format numerik (float) agar dapat diproses lebih lanjut oleh model.

Selanjutnya, dilakukan proses normalisasi terhadap data menggunakan StandardScaler. Normalisasi ini penting untuk menyamakan skala antar nilai dan membantu proses pelatihan model menjadi lebih stabil dan cepat konvergen. Setelah data ternormalisasi, dibentuk struktur data sekuensial menggunakan pendekatan sliding window. Dalam pendekatan ini, selama pelatihan model akan menggunakan data historis selama 30 hari (window_size = 30) untuk memprediksi konsumsi energi satu hari ke depan. Proses ini dilakukan secara iteratif untuk menghasilkan ratusan pasangan *input-output*.

Dataset kemudian dibagi menjadi dua bagian: 83% digunakan untuk data latih (*training set*) dan sisanya sebesar 17% digunakan sebagai data uji (*test set*). Proporsi ini

ditetapkan untuk tidak hanya mempertahankan keseimbangan antara jumlah data pelatihan dan pengujian, tetapi juga berdasarkan analisis visual pola data. Sekitar 83% dari total data menunjukkan adanya perubahan penting, yaitu awal dari kenaikan tren baru setelah fase penurunan. Dengan demikian, pemisahan data dilakukan di titik tersebut, memungkinkan model untuk memahami perubahan tren yang signifikan dan menguji kemampuannya dalam meramalkan pergerakan setelah adanya perubahan itu.

b) Arsitektur Model LSTM

Arsitektur model LSTM yang digunakan terdiri dari beberapa lapisan utama. Lapisan pertama adalah *layer input* yang menyesuaikan dimensi input berdasarkan *window size* dan jumlah fitur yang digunakan (dalam hal ini satu fitur yaitu GWh). *Layer* berikutnya adalah dua buah *layer* LSTM. *Layer* LSTM pertama memiliki jumlah unit neuron yang bervariasi, hasil dari tuning hyperparameter, sedangkan *layer* LSTM kedua memiliki jumlah unit yang lebih kecil dan berfungsi memperkuat representasi temporal dari *layer* sebelumnya.

Setelah layer LSTM, digunakan satu *layer dropout*. Fungsi dari dropout adalah untuk mengurangi kemungkinan *overfitting* dengan cara mengabaikan sebagian neuron secara acak selama proses pelatihan. Terakhir, digunakan *layer dense* atau *fully connected* yang menghasilkan output prediksi berupa nilai GWh untuk satu hari ke depan.

Model dikompilasi menggunakan fungsi *loss* berupa Mean Squared Error (MSE) karena tugas yang dilakukan adalah regresi, dan MSE sensitif terhadap error besar. *Optimizer* yang digunakan adalah Adam, yang merupakan *optimizer* adaptif dan umum digunakan dalam *deep learning* karena konvergensi cepat serta performa yang stabil.

c) Hyperparameter Tuning dengan Keras Tuner

Untuk mencapai performa model yang maksimal, dilakukan *tuning* hyperparameter dengan Keras Tuner. Tujuan dari langkah ini adalah menemukan kombinasi parameter yang optimal agar model mampu belajar dengan lebih efektif dan memberikan hasil prediksi yang tepat. Tiga parameter kunci yang dioptimalkan meliputi jumlah unit neuron di *layer* LSTM pertama dan kedua, serta tingkat *dropout*.

Unit pertama (units1) menetapkan kapasitas untuk merepresentasikan pola sekuensial pada *layer* LSTM pertama, sedangkan unit kedua (units2) meningkatkan pemahaman terhadap pola temporal di *layer* setelahnya. Selain itu, *dropout* diterapkan sebagai teknik regularisasi untuk mencegah terjadinya *overfitting*.

Dalam proses penyesuaian, Keras Tuner mengeksplor berbagai kombinasi parameter dengan menggunakan metode pencarian serta evaluasi yang didasarkan pada nilai *loss* validasi (val_loss). Proses ini berlangsung sekitar 14 menit dan 5 detik, dan menghasilkan kombinasi parameter terbaik pada tabel 4.6 sebagai berikut:

**Tabel 4. 6** Kombinasi Parameter Terbaik dengan Tuning

| Hyperparameter | Nilai Terbaik |
|---|---|
| units1 | 96 |
| units2 | 48 |
| dropout | 0.2 |
| val_loss | 0.00103 |

Nilai val_loss terkecil yang diperoleh menunjukkan bahwa kombinasi parameter tersebut memberikan performa terbaik selama proses validasi. Dengan konfigurasi ini, model memiliki keseimbangan antara kompleksitas dan kemampuan generalisasi terhadap data yang belum pernah dilihat sebelumnya.Nilai val_loss terkecil yang diperoleh menunjukkan bahwa kombinasi parameter tersebut memberikan performa terbaik selama

proses validasi. Dengan konfigurasi ini, model memiliki keseimbangan antara kompleksitas dan kemampuan generalisasi terhadap data yang belum pernah dilihat sebelumnya.

d) Hasil Pelatihan Model

Setelah diperoleh arsitektur dan hyperparameter terbaik, model LSTM kemudian dilatih menggunakan data latih selama 30 *epoch* dengan *batch size* sebesar 16. Selama pelatihan, 10% dari data latih digunakan sebagai data validasi untuk memantau performa model terhadap data yang belum pernah dilihat.

Hasil pelatihan menunjukkan bahwa nilai *loss* dan val_loss mengalami penurunan signifikan pada epoch-epoch awal, yang mengindikasikan bahwa model berhasil belajar mengenali pola dalam data. Setelah beberapa epoch, penurunan val_loss mulai melambat dan cenderung stabil, menunjukkan bahwa model berada dalam kondisi optimal dan tidak mengalami *overfitting* yang berlebihan. Berikut adalah *output* hasil pelatihan model:

**Tabel 4. 7** Hasil Pelatihan Model

| Epoch | Loss | Val_Loss |
|---|---|---|
| 1 | 0.0500 | 0.0200 |
| 2 | 0.0343 | 0.0059 |
| 3 | 0.0299 | 0.0129 |
| ⋮ | ⋮ | ⋮ |
| 30 | 0.0424 | 0.0137 |

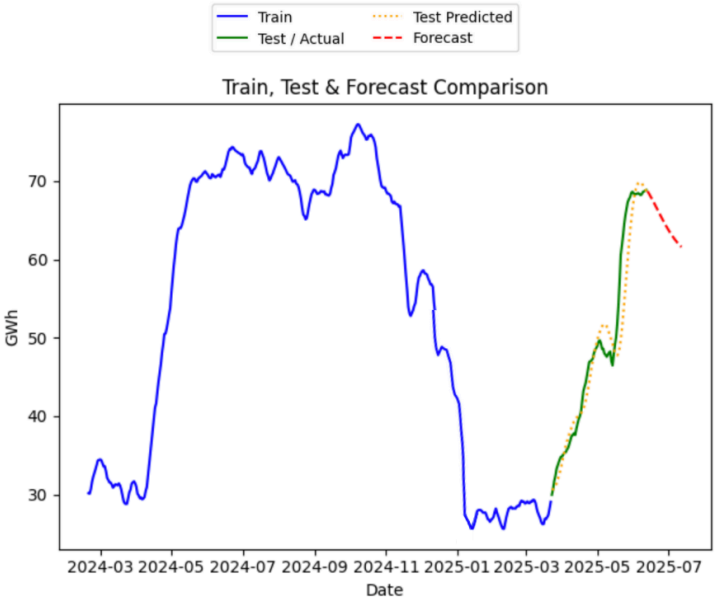
Secara keseluruhan, proses pelatihan menunjukkan bahwa model LSTM mampu belajar dengan baik dari data historis, dan memiliki performa yang cukup baik terhadap data validasi. Hasil ini memberikan dasar yang kuat untuk melanjutkan ke tahap evaluasi dan prediksi pada data uji serta *forecasting* ke depan.

e) Evaluasi Model dan Hasil *Forecasting*

Setelah model selesai dilatih, langkah selanjutnya adalah mengevaluasi performanya terhadap data uji serta menghasilkan prediksi konsumsi energi untuk 30 hari ke depan. Evaluasi dilakukan dengan membandingkan hasil prediksi terhadap data aktual.

Pertama, model digunakan untuk memprediksi nilai konsumsi energi pada data latih dan uji. Hasil prediksi ini masih dalam bentuk data yang telah dinormalisasi. Oleh karena itu, dilakukan proses *inverse transform* menggunakan fungsi inverse_transform dari StandardScaler untuk mengembalikan nilai prediksi dan aktual ke skala semula (GWh), sehingga dapat dilakukan perbandingan secara langsung. Setelah itu, dilakukan proses *forecasting* selama 30 hari ke depan. *Forecast* ini dilakukan dengan menggunakan pendekatan *autoregressive*, yaitu prediksi pada hari ke-1 digunakan sebagai *input* untuk prediksi hari ke-2, dan seterusnya. Nilai-nilai hasil prediksi tersebut juga dikembalikan ke skala asli dengan metode denormalisasi.

Untuk memvisualisasikan hasil, digunakan grafik garis yang membandingkan tiga kelompok data: (1) data pelatihan (*Train*), (2) data uji beserta prediksinya (*Test dan Test Predicted*), serta (3) hasil *forecasting* selama 30 hari ke depan (*Forecast*) pada Gambar 4.5 berikut.




**Gambar 4. 6** Visualisasi Hasil

Grafik ini memberikan gambaran yang komprehensif terhadap performa model baik dalam mempelajari pola historis maupun dalam melakukan prediksi masa depan.

f) Evaluasi Akurasi Model

Setelah proses pelatihan dan prediksi dilakukan, model LSTM dievaluasi berdasarkan tiga metrik utama: Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), dan Mean Absolute Percentage Error (MAPE). Evaluasi ini dilakukan pada dua set data, yaitu data latih (*train*) dan data uji (*test*), guna melihat seberapa baik model dalam mempelajari data historis dan seberapa baik model dapat menggeneralisasi terhadap data baru yang belum pernah dilihat sebelumnya. Tabel 4.8 berikut menunjukkan hasil evaluasi model.

**Tabel 4.7** Hasil Evaluasi Model

| Metrik Evaluasi | Data *Train* | Data *Test* |
|---|---|---|
| MAE | 1.4772 | 0.9869 |
| RMSE | 1.6531 | 1.2492 |
| MAPE | 3.45% | 2.12% |

Berdasarkan tabel tersebut, nilai MAE dan RMSE pada data *test* lebih rendah dibandingkan dengan data *train*, yang menunjukkan bahwa model tidak mengalami *overfitting* dan justru dapat mengeneralisasi dengan baik. Nilai MAPE yang rendah (<5%) baik pada data latih maupun data uji juga mengindikasikan bahwa model memiliki tingkat kesalahan prediksi yang relatif kecil terhadap nilai sebenarnya, dan dapat dianggap sangat baik dalam konteks data time series energi.

Secara keseluruhan, model LSTM yang digunakan telah menunjukkan performa prediksi yang cukup akurat dan stabil, baik untuk data yang digunakan dalam pelatihan maupun data yang digunakan untuk pengujian.

## 4.6 Perbandingan Kinerja Model

Dalam penelitian ini dilakukan perbandingan performa antara dua pendekatan pemodelan *time series*, yaitu ARIMA sebagai model linear dan LSTM sebagai model non-linear berbasis *deep learning*. Tujuan dari perbandingan ini adalah untuk menilai model mana

yang lebih baik dalam memprediksi konsumsi energi sektor minyak di Brasil berdasarkan pola historis. Berikut adalah Tabel 4.8 mencakup perbandingan akurasi model ARIMA dan LSTM.

**Tabel 4. 8** Perbandingan Akurasi Model ARIMA dan LSTM

| **Metrik Evaluasi** | **ARIMA (*test*)** | **LSTM (Data *train*)** | **LSTM (Data *test*)** |
|---|---|---|---|
| MAE | 20,4873 | 1.4772 | 0.9869 |
| RMSE | 24.4954 | 1.6531 | 1.2492 |
| MAPE | 38.26% | 3.45% | 2.12% |

Berdasarkan hasil evaluasi tersebut, terlihat bahwa model LSTM secara signifikan mengungguli ARIMA pada semua metrik evaluasi. Nilai MAE dan RMSE LSTM jauh lebih rendah, menunjukkan bahwa kesalahan absolut dan kuadrat rata-rata prediksinya jauh lebih kecil dibanding ARIMA. Demikian pula, nilai MAPE yang jauh lebih rendah (di bawah 5%) menunjukkan bahwa model LSTM memiliki akurasi prediksi yang tinggi dan kesalahan relatif yang kecil, bahkan pada data uji yang belum pernah dilihat sebelumnya.

Selain itu, performa LSTM yang baik pada data uji menunjukkan bahwa model ini memiliki kemampuan generalisasi yang kuat dan tidak mengalami *overfitting*. Sebaliknya, ARIMA yang berbasis linear kurang mampu menangkap kompleksitas pola non-linear yang ada pada data konsumsi energi, sehingga menghasilkan kesalahan prediksi yang cukup besar.

Dengan demikian, dapat disimpulkan bahwa LSTM lebih unggul dan lebih cocok digunakan dalam konteks peramalan konsumsi energi minyak di Brasil, terutama ketika data menunjukkan karakteristik non-linear dan fluktuatif. Model ARIMA dapat tetap digunakan sebagai model pembanding atau *baseline*, namun performanya cenderung terbatas pada pola data yang bersifat lebih stasioner dan linier.

# BAB 5
# KESIMPULAN DAN SARAN

## 5.1 Kesimpulan

Berdasarkan hasil analisis data pada kasus data GWh Minyak di Brasil didapatkan beberapa kesimpulan sebagai berikut.

1. Data konsumsi energi sektor minyak Brasil menunjukkan pola yang tidak stasioner dan fluktuasi yang cukup ekstrim tanpa pola musiman yang tetap. Uji Teraesvirta Neural Network Test menghasilkan p-value 0,04242 (< 0,05), sehingga data dinyatakan memiliki karakteristik non-linier. Hal ini mendukung penggunaan model non-linier seperti LSTM dalam penelitian ini.
2. Model ARIMA dengan parameter (3,1,4) diterapkan setelah data dibuat stasioner dengan differencing. Model ini menunjukkan keterbatasan dalam menangkap fluktuasi mendadak dan pola musiman pada data. Evaluasi pada data uji menghasilkan nilai error yang relatif tinggi, yaitu: MAE tidak tersedia secara eksplisit, namun RMSE sebesar 24.4954 dan MAPE sebesar 38.26%, yang menunjukkan kesalahan prediksi yang cukup besar1.
3. Model LSTM yang dibangun dengan dua lapisan LSTM dan dropout 0.2, serta tuning hyperparameter menggunakan Keras Tuner, menunjukkan performa yang jauh lebih baik. Evaluasi pada data latih dan uji menghasilkan:
    - MAE: 1.4772 (data latih) dan 0.9869 (data uji)
    - RMSE: 1.6531 (data latih) dan 1.2492 (data uji)
    - MAPE: 3.45% (data latih) dan 2.12% (data uji)

    Nilai-nilai ini jauh lebih rendah dibandingkan ARIMA, menandakan prediksi yang lebih akurat dan konsisten
4. Perbedaan signifikan dalam metrik evaluasi menunjukkan bahwa LSTM lebih mampu menangkap pola non-linier dan dinamika kompleks dalam data konsumsi energi minyak Brasil dibandingkan ARIMA yang berbasis linear. Hal ini juga didukung oleh proses pelatihan LSTM yang menunjukkan penurunan loss dan val_loss yang stabil, menandakan model tidak mengalami overfitting dan memiliki kemampuan generalisasi yang baik.
5. Dengan performa yang superior, model LSTM direkomendasikan sebagai pendekatan utama untuk prediksi konsumsi energi sektor minyak di Brasil, terutama ketika data menunjukkan karakteristik non-linier dan fluktuasi yang kompleks. Model ARIMA dapat digunakan sebagai baseline, namun kurang efektif untuk data dengan pola seperti ini.

## 5.2 Saran

Saran untuk penelitian selanjutnya adalah melakukan penyesuaian atau eksplorasi lebih lanjut terhadap hyperparameter yang digunakan, seperti jumlah unit pada lapisan LSTM, panjang timestep (timelag), serta struktur arsitektur model LSTM secara keseluruhan. Selain itu, dapat juga dipertimbangkan penggunaan model hybrid yang menggabungkan pendekatan linear dan non-linear, misalnya ARIMA-LSTM, untuk menangkap dinamika data yang lebih kompleks. Penyesuaian ini bertujuan agar model tidak mengalami *overfitting* atau *underfitting* terhadap data latih dan tetap mempertahankan

performa yang baik saat diterapkan pada data uji. Dengan demikian, akurasi dan kemampuan generalisasi model dalam memprediksi konsumsi energi dapat terus ditingkatkan.

# DAFTAR PUSTAKA


Arissinta, I. O., Sulistiyawati, I. D., Kurniyanto, D., & Kharisudin, I. (2022). Pemodelan Time Series untuk Peramalan Web Traffic Menggunakan Algoritma ARIMA, LSTM, dan GRU. PRISMA, Prosiding Seminar Nasional Matematika, 5, 693–700. https://journal.unnes.ac.id/sju/index.php/prisma/

Nurul Janah. (2024). Perbandingan metode LSTM dan ARIMA dalam memprediksi tingkat inflasi di Provinsi Kepulauan Bangka Belitung [Skripsi, Universitas Bangka Belitung]. Repositori Universitas Bangka Belitung. https://repository.ubb.ac.id/id/eprint/9908/

Priambodo, A. Z., & Mahmudah. (2020). Prediction Model for the Number of ARI Cases in Children in Surabaya Using ARIMA Method. Jurnal Biometrika dan Kependudukan, 9(1), 18–26. https://doi.org/10.20473/jbk.v9i1.2020.18-26

Sadya, S. (2022). *Peramalan Tingkat Pengangguran di Indonesia Menggunakan Support Vector Regression dan Long Short-Term Memory* [Undergraduate thesis, Institut Teknologi Sepuluh Nopember]. Institut Teknologi Sepuluh Nopember Repository. https://repository.its.ac.id/94542/1/06211840000092-Undergraduate_Thesis.pdf

Sautomo, S., & Pardede, H. F. (2021). Prediksi Belanja Pemerintah Indonesia Menggunakan Long Short-Term Memory (LSTM). Jurnal RESTI (Rekayasa Sistem dan Teknologi Informasi), 5(1), 99–106. https://doi.org/10.29207/resti.v5i1.2815

Sunendar, N., Putro, H. P., & Hesananda, R. (2025). Prediksi Penjualan Aerosol Menggunakan Algoritma ARIMA, LSTM dan GRU. INSOLOGI: Jurnal Sains dan Teknologi, 4(1), 113–126. https://doi.org/10.55123/insologi.v4i1.4868

Tussifah, S. A. (2022). Analisis perbandingan kinerja model ARIMA, LSTM dan GRU pada prediksi data time series [Skripsi, Universitas Islam Negeri Jakarta]. Repositori UIN Jakarta. https://repository.uinjkt.ac.id/dspace/bitstream/123456789/68556/1/SUCI%20AMALIA%20TUSSIFAH-FST.pdf

Wei, W. W. S. (2006). Time Series Analysis: Univariate and Multivariate Methods (2nd ed.). Upper Saddle River, NJ: Pearson-Addison Wesley.

Wibawa, T. S. (2021). Pemodelan hybrid ARIMA-LSTM dalam meramalkan harga saham (Studi kasus: Harga penutupan saham Bank BCA) [Skripsi, Universitas Brawijaya]. Repositori Universitas Brawijaya. http://repository.ub.ac.id/id/eprint/184941

Wibowo, M. A. (2020). Penerapan metode deep learning Long Short-Term Memory Network pada prediksi data time-series [Tugas Akhir, Institut Teknologi Sepuluh Nopember]. Repositori Institut Teknologi Sepuluh Nopember. http://repository.its.ac.id/id/eprint/77845

# LAMPIRAN

**DATA**
**https://its.id/m/DatasetFPKel12_GWhBrazil**

**OUTPUT UJI NON LINEARITAS**

```
        Teraesvirta Neural Network Test

data:  gwh_ts
X-squared = 6.3201, df = 2, p-value = 0.04242
```

**SYNTAX R STUDIO**

```
# ==========================================
# 1. Install & Load Packages
# ==========================================
# install.packages(c("forecast", "tseries", "ggplot2"))
library(forecast)
library(tseries)
library(ggplot2)
# ==========================================
# 2. Load & Preprocess Data
# ==========================================
data <- read.csv("E:/Semester 6/SML-A/BRAZIL.csv", sep = ";")
# Ubah angka dan tanggal
data$GWh <- as.numeric(gsub(",", ".", data$GWh))
data$date <- as.Date(data$date, format = "%d/%m/%Y")
# Pengecekan Missing Value sebelum dihapus
missing_count <- sum(is.na(data$GWh))
cat("Jumlah missing value pada kolom GWh:", missing_count, "\n")
# Jika ada missing value, hapus baris tersebut
if (missing_count > 0) {
```

```
  data <- na.omit(data)
  cat("Missing value telah dihapus. Data sekarang memiliki", nrow(data), "baris.\n")
} else {
  cat("Tidak ditemukan missing value pada data.\n")
}
# Buat time series object (harian)
ts_data <- ts(data$GWh, frequency = 365)
# ==========================================
# 3. Periksa Stasioneritas (Original Data)
# ==========================================
adf_result <- adf.test(ts_data)
cat("ADF p-value (Original):", adf_result$p.value, "\n")
# ==========================================
# 4. Differencing (Jika Tidak Stasioner)
# ==========================================
if (adf_result$p.value > 0.05) {
  ts_diff <- diff(ts_data)
  adf_diff <- adf.test(ts_diff)
  cat("ADF p-value (Setelah Differencing):", adf_diff$p.value, "\n")
  # Visualisasi setelah differencing
  autoplot(ts_diff) + ggtitle("Data Setelah Differencing") +
    ylab("GWh") + xlab("Tanggal")
  ggAcf(ts_diff) + ggtitle("ACF Data Differencing")
  ggPacf(ts_diff) + ggtitle("PACF Data Differencing")
}
# ==========================================
# 5. Split Train & Test Data
# ==========================================
n <- length(ts_data)
n_train <- floor(0.83 * n)
n_test <- n - n_train
```

```
ts_train <- ts(ts_data[1:n_train], start = start(ts_data), frequency = frequency(ts_data))

ts_test <- ts(ts_data[(n_train + 1):n], start = time(ts_data)[n_train + 1], frequency =
frequency(ts_data))

# ========================================

# 6. ARIMA (3,1,4)

# ========================================

lambda <- BoxCox.lambda(ts_train)

lambda

model_arima_fixed <- Arima(ts_train, order = c(3, 1, 4), lambda = lambda)

summary(model_arima_fixed)

# Evaluasi Model

cat("AIC ARIMA (3,1,4):", AIC(model_arima_fixed), "\n")

# Forecast pada test data

forecast_test_fixed <- forecast(model_arima_fixed, h = n_test)

# Hitung evaluasi: MAPE, MAE, RMSE

y_test_pred <- as.numeric(forecast_test_fixed$mean)

y_test_actual <- as.numeric(ts_test)

valid_idx <- !is.na(y_test_pred) & !is.na(y_test_actual)

y_test_pred <- y_test_pred[valid_idx]

y_test_actual <- y_test_actual[valid_idx]

mape <- mean(abs((y_test_actual - y_test_pred) / y_test_actual)) * 100

mae <- mean(abs(y_test_actual - y_test_pred))

rmse <- sqrt(mean((y_test_actual - y_test_pred)^2))

cat("MAPE Test:", round(mape, 4), "%\n")

cat("MAE Test:", round(mae, 4), "\n")

cat("RMSE Test:", round(rmse, 4), "\n")

# ========================================

# 7. Visualisasi Forecast

# ========================================

autoplot(ts_data) +

  autolayer(forecast_test_fixed, series = "Forecast (ARIMA (3,1,4))", PI = FALSE) +
```

```
  autolayer(ts_test, series = "Test Data") +
  ggtitle("Forecast vs Test Data (ARIMA (3,1,4))") +
  ylab("GWh") + xlab("Tanggal")
# ==========================================
# 8. Final Forecast (Full Data)
# ==========================================
model_final <- Arima(ts_data, order = c(3, 1, 4), lambda = lambda)
forecast_final <- forecast(model_final, h = 30)
# Visualisasi Final Forecast
autoplot(forecast_final) +
  ggtitle("Forecast Konsumsi Minyak 30 Hari ke Depan (ARIMA (3,1,4))") +
  ylab("GWh") + xlab("Tanggal")
# ==========================================
# 9. Cek Residual
# ==========================================
# Residual Analysis
checkresiduals(model_arima_fixed)
# Uji Normalitas Residual
shapiro_test <- shapiro.test(residuals(model_arima_fixed))
cat("Shapiro-Wilk Test p-value:", shapiro_test$p.value, "\n")
if (shapiro_test$p.value > 0.05) {
  cat("Residuals mengikuti distribusi normal.\n")
} else {
  cat("Residuals tidak mengikuti distribusi normal.\n")
}
# Visualisasi Residual
autoplot(residuals(model_arima_fixed)) +
  ggtitle("Residual Plot (ARIMA (3,1,4))") +
  ylab("Residual") + xlab("Waktu")
# Uji Autocorrelation
Box.test(residuals(model_arima_fixed), lag = 10, type = "Ljung-Box")
```

```
cat("Ljung-Box Test (p-value):", Box.test(residuals(model_arima_fixed), lag = 10)$p.value, "\n")
if (Box.test(residuals(model_arima_fixed), lag = 10)$p.value > 0.05) {
  cat("Residuals tidak memiliki autocorrelation signifikan.\n")
} else {
  cat("Residuals memiliki autocorrelation signifikan.\n")
}
```

**SYNTAX GOOGLE COLAB**

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Dropout
from keras_tuner.tuners import RandomSearch
from google.colab import drive
drive.mount('/content/drive')
path = "/content/drive/MyDrive/DATASET FP/YANG INI BRAZIL.xlsx"
df = pd.read_excel(path)
df
```

```
df['date'] = pd.to_datetime(df['date'], format='%d/%m/%Y')
df = df.sort_values('date').reset_index(drop=True)
values = df['GWh'].values.reshape(-1, 1).astype(float)
scaler = StandardScaler()
scaled_values = scaler.fit_transform(values)
```

```
def create_dataset(data, window_size, next_predict=1):
    X, y = [], []
    for i in range(window_size, len(data) - next_predict +1):
        X.append(data[i - window_size:i])
        y.append(data[i + next_predict -1:i + next_predict, 0])
    return np.array(X), np.array(y)
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Input, LSTM, Dense, Dropout

def build_model(hp, input_shape):
    model = Sequential()
    model.add(Input(shape=input_shape))  # <-- Tambahkan layer Input di
sini
    model.add(LSTM(units=hp.Int('units1', 32, 128, step=32),
                   activation='relu',
                   return_sequences=True))
    model.add(LSTM(units=hp.Int('units2', 16, 64, step=16),
```

```
                    activation='relu',
                    return_sequences=False))
    model.add(Dropout(hp.Float('dropout', 0.1, 0.5, step=0.1)))
    model.add(Dense(1))
    model.compile(optimizer='adam', loss='mse')
    return model
```

```
window_sizes = [15, 30, 50]

results = {}

for window_size in window_sizes:
    print(f"\n=== Window Size: {window_size} ===")

    # Buat dataset
    X, y = create_dataset(scaled_values, window_size)
```

```
split = int(len(X) * 0.83)
X_train, X_test = X[:split], X[split:]
y_train, y_test = y[:split], y[split:]
```

```
tuner = RandomSearch(
        lambda hp: build_model(hp, input_shape=(X_train.shape[1],
X_train.shape[2])),
        objective='val_loss',
        max_trials=10,
        executions_per_trial=1,
        directory='tuning_lstm',
        project_name=f'forecast_GWh_ws{window_size}',
        overwrite=True
    )

tuner.search(X_train, y_train, epochs=50, batch_size=16,
validation_split=0.1, verbose=1)

best_model = tuner.get_best_models(num_models=1)[0]
history = best_model.fit(X_train, y_train, epochs=30, batch_size=16,
validation_split=0.1, verbose=1)

y_train_pred = best_model.predict(X_train)
y_test_pred = best_model.predict(X_test)
y_train_inv = scaler.inverse_transform(y_train)
y_test_inv = scaler.inverse_transform(y_test)
y_train_pred_inv = scaler.inverse_transform(y_train_pred)
y_test_pred_inv = scaler.inverse_transform(y_test_pred)

input_seq = X[-1]
```

```
future_preds = []
for _ in range(30):
    pred = best_model.predict(input_seq.reshape(1, window_size, 1),
verbose=0)
    future_preds.append(pred[0][0])
    input_seq = np.append(input_seq[1:], [[pred[0][0]]], axis=0)

future_preds_inv =
scaler.inverse_transform(np.array(future_preds).reshape(-1, 1))

# Plot Train, Test, dan Forecast dalam Satu Grafik
plt.figure(figsize=(14, 6))

dates = df['date']
train_dates = dates[window_size:split + window_size]
test_dates = dates[split + window_size:]

# Plot Train
plt.plot(train_dates, y_train_inv, label='Train', color='blue')
# Plot Test
plt.plot(test_dates, y_test_inv, label='Test / Actual', color='green')
plt.plot(test_dates, y_test_pred_inv, label='Test Predicted',
color='orange', linestyle='dotted')
# Plot Forecast
future_dates = pd.date_range(start=test_dates.iloc[-1] +
pd.Timedelta(days=1), periods=30, freq='D')
plt.plot(future_dates, future_preds_inv, label='Forecast', color='red',
linestyle='dashed')

plt.xlabel('Date')
plt.ylabel('GWh')
plt.title('Train, Test & Forecast Comparison')
plt.legend()
plt.tight_layout()
plt.show()

from sklearn.metrics import mean_absolute_error, mean_squared_error,
mean_absolute_percentage_error
import numpy as np

# Misal y_train_inv, y_train_pred_inv, y_test_inv, y_test_pred_inv sudah
tersedia dari proses sebelumnya
# y_train_inv = actual train data (inverse scaled)
# y_train_pred_inv = prediksi train (inverse scaled)
# y_test_inv = actual test data (inverse scaled)
# y_test_pred_inv = prediksi test (inverse scaled)

# Hitung MAE
mae_train = mean_absolute_error(y_train_inv, y_train_pred_inv)
mae_test = mean_absolute_error(y_test_inv, y_test_pred_inv)
```

```
# Hitung RMSE
rmse_train = np.sqrt(mean_squared_error(y_train_inv, y_train_pred_inv))
rmse_test = np.sqrt(mean_squared_error(y_test_inv, y_test_pred_inv))

# Hitung MAPE
mape_train = mean_absolute_percentage_error(y_train_inv, y_train_pred_inv)
* 100  # dalam persen
mape_test = mean_absolute_percentage_error(y_test_inv, y_test_pred_inv) *
100

print(f"Train MAE: {mae_train:.4f}")
print(f"Test MAE: {mae_test:.4f}")
print(f"Train RMSE: {rmse_train:.4f}")
print(f"Test RMSE: {rmse_test:.4f}")
print(f"Train MAPE: {mape_train:.2f}%")
print(f"Test MAPE: {mape_test:.2f}%")

import numpy as np
import matplotlib.pyplot as plt
from statsmodels.stats.diagnostic import acorr_ljungbox, het_arch
from scipy.stats import shapiro

# Hitung residual
residuals = y_test_inv.flatten() - y_test_pred_inv.flatten()

# Plot residual
plt.figure(figsize=(10,4))
plt.plot(residuals)
plt.title('Residual Plot')
plt.xlabel('Index')
plt.ylabel('Residual')
plt.grid(True)
plt.show()

# Uji Autokorelasi (Ljung-Box)
lb_test = acorr_ljungbox(residuals, lags=[10], return_df=True)
print("Ljung-Box test:\n", lb_test)

# Uji Normalitas (Shapiro-Wilk)
shapiro_test = shapiro(residuals)
print(f"Shapiro-Wilk test statistic={shapiro_test.statistic:.4f}, p-
value={shapiro_test.pvalue:.4f}")

# Uji Heteroskedastisitas (ARCH test)
arch_test = het_arch(residuals)
print(f"ARCH test statistic={arch_test[0]:.4f}, p-
value={arch_test[1]:.4f}")
```

**OUTPUT R STUDIO**

1. Prepocessing Data

```
> missing_count <- sum(is.na(data$GWh))
> cat("Jumlah missing value pada kolom GWh:", missing_co
Jumlah missing value pada kolom GWh: 0
> # Jika ada missing value, hapus baris tersebut
> if (missing_count > 0) {
+   data <- na.omit(data)
+   cat("Missing value telah dihapus. Data sekarang memi
+ } else {
+   cat("Tidak ditemukan missing value pada data.\n")
+ }
Tidak ditemukan missing value pada data.
```

2. Uji Stasioneritas - ADF Test

```
> # Buat time series object (harian)
> ts_data <- ts(data$GWh, frequency = 365)
> # ========================================
> # 3. Periksa Stasioneritas (Original Data)
> # ========================================
> adf_result <- adf.test(ts_data)
> cat("ADF p-value (Original):", adf_result$p.value, "\n")
ADF p-value (Original): 0.6957036
```

3. Differencing

```
+   # Visualisasi setelah differencing
+   autoplot(ts_diff) + ggtitle("Data Setelah Differencing") +
+     ylab("GWh") + xlab("Tanggal")
+   ggAcf(ts_diff) + ggtitle("ACF Data Differencing")
+   ggPacf(ts_diff) + ggtitle("PACF Data Differencing")
+ }
ADF p-value (Setelah Differencing): 0.01
```

4. Box Cox

```
> # ========================================
> # 6. ARIMA (3,1,4)
> # ========================================
> lambda <- BoxCox.lambda(ts_train)
> lambda
[1] 1
```

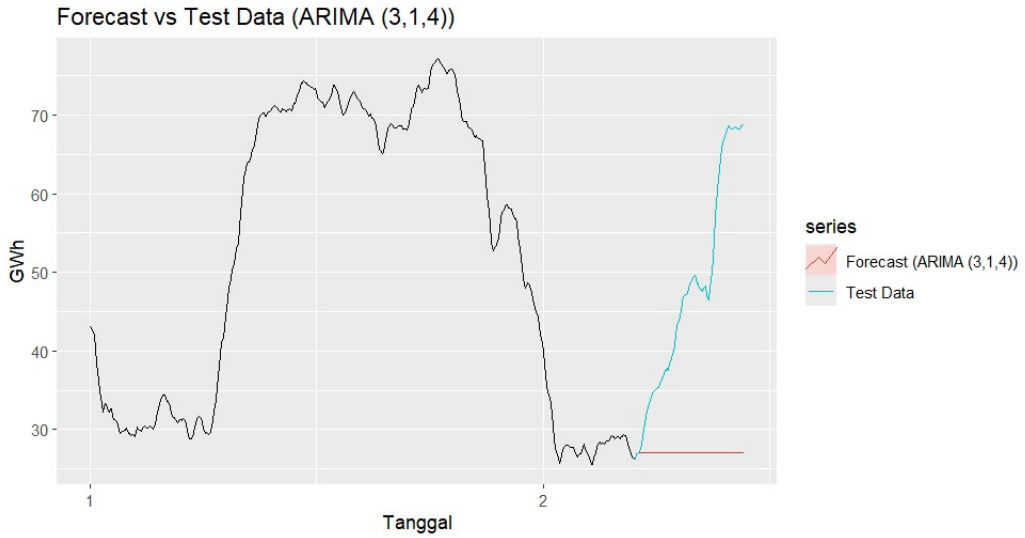
5. Best Model ARIMA

```
Series: ts_train
ARIMA(3,1,4)
Box Cox transformation: lambda= 1

Coefficients:
         ar1     ar2      ar3     ma1      ma2     ma3     ma4
      0.3440  0.4471  -0.3477  0.5485  -0.2647  0.4790  0.8923
s.e.  0.0513  0.0488   0.0518  0.0290   0.0280  0.0401  0.0347

sigma^2 = 0.1074:  log likelihood = -132.63
AIC=281.26   AICc=281.59   BIC=313.9

Training set error measures:
                      ME      RMSE       MAE         MPE      MAPE       MASE
Training set -0.007360761 0.3247296 0.2378754 -0.0136653 0.5421698 0.06477723
                    ACF1
Training set -0.01066867
```

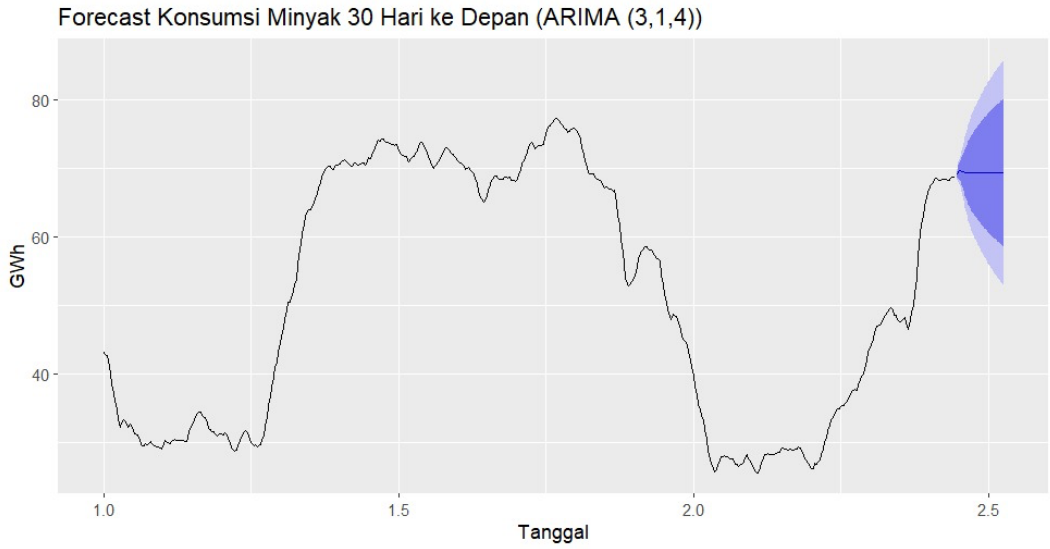
6. Akurasi

```
AIC ARIMA (3,1,4): 281.2576
> # Forecast pada test data
> forecast_test_fixed <- forecast(model_arima_fixed,
> # Hitung evaluasi: MAPE, MAE, RMSE
> y_test_pred <- as.numeric(forecast_test_fixed$mean)
> y_test_actual <- as.numeric(ts_test)
> valid_idx <- !is.na(y_test_pred) & !is.na(y_test_ac
> y_test_pred <- y_test_pred[valid_idx]
> y_test_actual <- y_test_actual[valid_idx]
> mape <- mean(abs((y_test_actual - y_test_pred) / y_
> mae <- mean(abs(y_test_actual - y_test_pred))
> rmse <- sqrt(mean((y_test_actual - y_test_pred)^2))
> cat("MAPE Test:", round(mape, 4), "%\n")
MAPE Test: 38.2596 %
> cat("MAE Test:", round(mae, 4), "\n")
MAE Test: 20.4873
> cat("RMSE Test:", round(rmse, 4), "\n")
RMSE Test: 24.4954
```

7. Plot

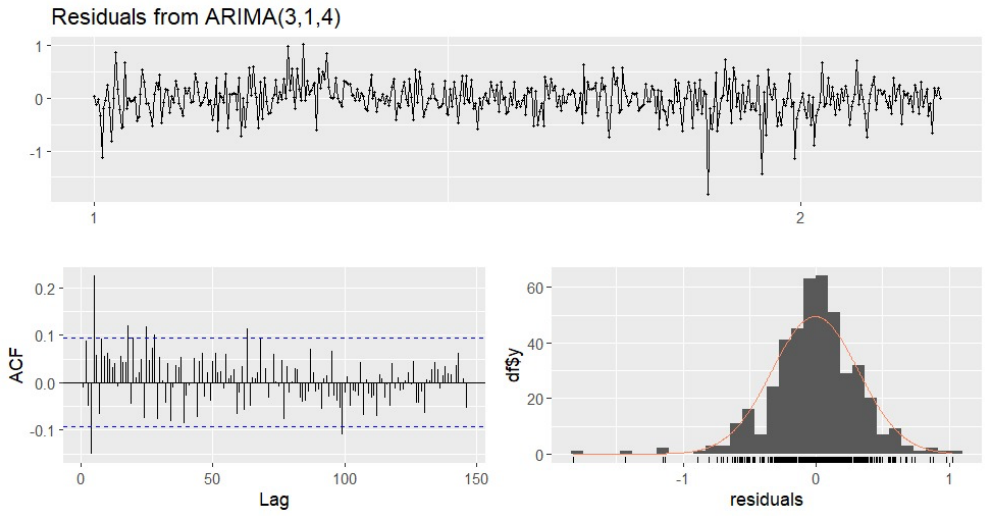



8. Plot Forecast



9. Asumsi Residual

```
> # Residual Analysis
> checkresiduals(model_arima_fixed)

        Ljung-Box test

data:  Residuals from ARIMA(3,1,4)
Q* = 144.69, df = 81, p-value = 1.768e-05

Model df: 7.   Total lags used: 88
```




**OUTPUT GOOGLE COLAB**

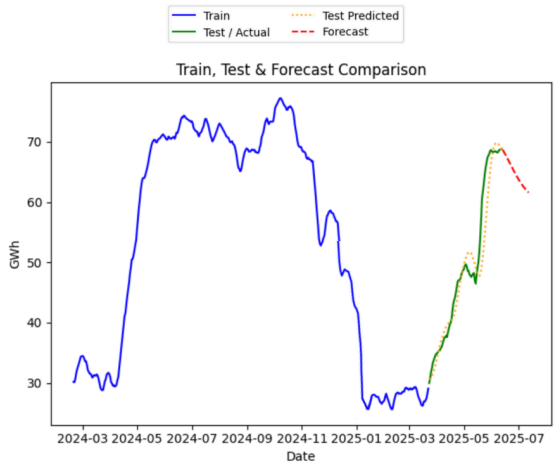
1. Window Size

```
=== Window Size: 15 ===

=== Window Size: 30 ===

=== Window Size: 50 ===
```

2. Tunning Hyperparameter

```
Trial 10 Complete [00h 01m 10s]
val_loss: 0.0019857638981193304

Best val_loss So Far: 0.0010336898267269135
Total elapsed time: 00h 14m 05s
```

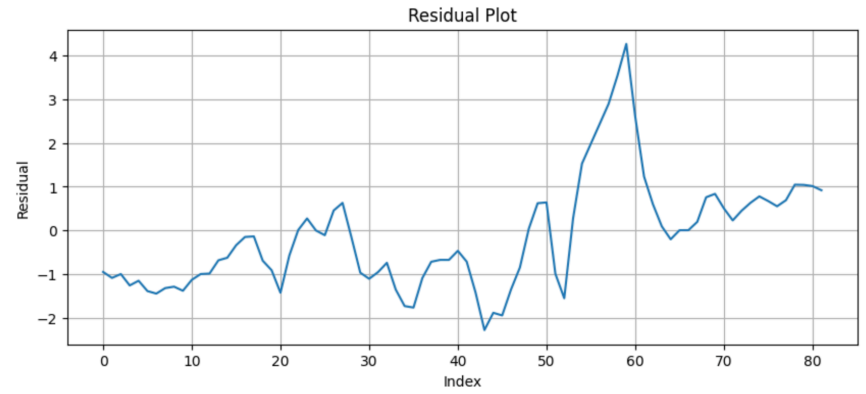
3. Plot




4. Akurasi

```
Train MAE: 1.4772
Test MAE: 0.9869
Train RMSE: 1.6531
Test RMSE: 1.2492
Train MAPE: 3.45%
Test MAPE: 2.12%
```

5. Asumsi Residual




```
Ljung-Box test:
        lb_stat     lb_pvalue
10  196.051065  1.074234e-36
Shapiro-Wilk test statistic=0.9254, p-value=0.0001
ARCH test statistic=47.2541, p-value=0.0000
```